

3/3/2 (Item 2 from file: 351)  
DIALOG(R) File 351: Derwent WPI  
(c) 2006 Thomson Derwent. All rts. reserv.

014190748 \*\*Image available\*\*  
WPI Acc No: 2002-011445/200201  
XRPX Acc No: N02-009461

**Job assigning method for parallel processing method includes parallel computer**

Patent Assignee: TAISHO PHARM CO LTD (TAIS ); HONDA MOTOR CO LTD (HOND );  
FUJI XEROX CO LTD (XERF ); INABATA S (INAB-I); KITAMURA K (KITA-I);  
MIYAKAWA N (MIYA-I); NAGASHIMA U (NAGA-I); TAKASHIMA H (TAKA-I); YAMADA S  
(YAMA-I)

Inventor: INABATA S; KITAMURA K; MIYAKAWA N; NAGASHIMA U; TAKASHIMA H;  
YAMADA S

Number of Countries: 096 Number of Patents: 008

**Patent Family:**

Patent No	Kind	Date	Applicat No	Kind	Date	Week
WO 200184343	A1	20011108	WO 2001JP3554	A	20010425	200201 B
JP 2001312485	A	20011109	JP 2000131253	A	20000428	200207
AU 200152561	A	20011112	AU 200152561	A	20010425	200222
EP 1286273	A1	20030226	EP 2001925891	A	20010425	200319
			WO 2001JP3554	A	20010425	
US 20030204576	A1	20031030	WO 2001JP3554	A	20010425	200372
			US 2002257913	A	20021018	
CN 1439130	A	20030827	CN 2001811717	A	20010425	200375
AU 771360	B2	20040318	AU 200152561	A	20010425	200454
CN 1655135	A	20050817	CN 2001811717	A	20010425	200572
			CN 20059542	A	20010425	

Priority Applications (No Type Date): JP 2000131253 A 20000428

**Patent Details:**

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
-----------	------	-----	----	----------	--------------

WO 200184343	A1	J	98	G06F-015/177	
--------------	----	---	----	--------------	--

Designated States (National): AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA  
CH CN CO CR CU CZ DE DK DM DZ EE ES FI GB GD GE GH GM HR HU ID IL IN IS  
KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ PL PT  
RO RU SD SE SG SI SK SL TJ TM TR TT TZ UA UG US UZ VN YU ZA ZW  
Designated States (Regional): AT BE CH CY DE DK EA ES FI FR GB GH GM GR  
IE IT KE LS LU MC MW MZ NL OA PT SD SE SL SZ TR TZ UG ZW

JP 2001312485	A		30	G06F-015/177	
---------------	---	--	----	--------------	--

AU 200152561	A				Based on patent WO 200184343
--------------	---	--	--	--	------------------------------

EP 1286273	A1	E		G06F-015/177	Based on patent WO 200184343
------------	----	---	--	--------------	------------------------------

Designated States (Regional): AL AT BE CH CY DE DK ES FI FR GB GR IE IT  
LI LT LU LV MC MK NL PT RO SE SI TR

US 20030204576	A1			G06F-015/177	
----------------	----	--	--	--------------	--

CN 1439130	A			G06F-015/177	
------------	---	--	--	--------------	--

AU 771360	B2			G06F-015/177	Previous Publ. patent AU 200152561
-----------	----	--	--	--------------	------------------------------------

Based on patent WO 200184343

CN 1655135	A			G06F-015/16	Div ex application CN 2001811717
------------	---	--	--	-------------	----------------------------------

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-312485

(43)Date of publication of application : 09.11.2001

(51)Int.Cl.

G06F 15/177

G06F 9/46

G06F 15/16

(21)Application number : 2000-131253

(71)Applicant : FUJI XEROX CO LTD  
TAISHO PHARMACEUT CO LTD

(22)Date of filing : 28.04.2000

(72)Inventor : YAMADA SOU  
INAHATA SHINJIROU  
MIYAGAWA NOBUAKI  
TAKASHIMA HAJIME  
KITAMURA KAZUYASU  
NAGASHIMA UNPEI

## (54) ALLOCATING METHOD FOR JOB BY PARALLEL PROCESSING METHOD AND PARALLEL PROCESSING METHOD

### (57)Abstract:

**PROBLEM TO BE SOLVED:** To efficiently allocate jobs to respective processors when parallel processing is performed by using a parallel computer constituted by connecting a host computer and plural processors by a common bus.

**SOLUTION:** A job whose ratio of a communication time and an operation time is specific or larger than several parts of that of a processor and a job which is smaller are allocated alternately to each processor. Plural processors and plural jobs are divided into plural groups while made to correspond to each other, jobs which are close in the size consisting of the communication time and operation time and jobs which are close in the ratio of the communication time and operation time are put in different job groups, and jobs which are close in the size consisting of the communication time and operation time and the ratio of the communication time and operation time are so allocated that the allocation places of the jobs in the job groups are mutually different among plural job groups.

### 待ち行列

0(0)
3 2(1)
1(2)
3 1(3)
2(4)
⋮
1 5(3 0)
1 7(3 1)
1 6(3 2)

## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision  
of rejection]

[Date of requesting appeal against examiner's  
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-312485 ✓  
(P2001-312485A)

(43) 公開日 平成13年11月9日 (2001.11.9)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード(参考)
G 0 6 F 15/177	6 7 4	G 0 6 F 15/177	6 7 4 C 5 B 0 4 5
9/46	3 6 0	9/46	3 6 0 B 5 B 0 9 8
15/16	6 1 0	15/16	6 1 0 B

審査請求 未請求 請求項の数14 O L (全 30 頁)

(21) 出願番号 特願2000-131253(P2000-131253)

(22) 出願日 平成12年4月28日(2000.4.28)

(71) 出願人 000005496

富士ゼロックス株式会社  
東京都港区赤坂二丁目17番22号

(71) 出願人 000002819

大正製薬株式会社  
東京都豊島区高田3丁目24番1号

(72) 発明者 山田 想

神奈川県足柄上郡中井町境430 グリーン  
テクなかい 富士ゼロックス株式会社内

(74) 代理人 100091546

弁理士 佐藤 正美

最終頁に続く

(54) 【発明の名称】 並列処理方法におけるジョブの割り当て方法および並列処理方法

(57) 【要約】

【課題】 ホスト計算機と、複数のプロセッサとが共通のバスにより接続されて構成される並列計算機を用いて並列処理を行う場合において、効率良く、各プロセッサにジョブの割り当てを行う。

【解決手段】 各プロセッサには、通信時間と演算時間との比が、所定の値あるいはプロセッサの数分の1よりも大きいジョブと、小さいジョブとを交互に割り当てる。または、複数のプロセッサと、複数のジョブとを、対応させて複数のグループに分割し、通信時間と演算時間とからなるサイズと、前記通信時間と演算時間との比とが近似するジョブは、ジョブ・グループの異なるものにそれぞれ属するようにし、かつ、通信時間と演算時間とからなるサイズと、通信時間と演算時間との比とが近似するジョブの、ジョブ・グループのそれぞれ内での割り当て順番が、複数のジョブ・グループの間において互いに異なるように割り当てる。

待ち行列

0(0)
3 2(1)
1(2)
3 1(3)
2(4)
⋮
1 5(3 0)
1 7(3 1)
1 6(3 2)

## 【特許請求の範囲】

【請求項 1】 ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなる複数のジョブを、前記複数のプロセッサにより並列に処理する並列処理方法において、

前記プロセッサのそれぞれには、前記ジョブの通信時間と演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 2】 請求項 1 に記載の並列処理方法におけるジョブの割り当て方法において、前記複数のジョブの全体の通信時間と、前記複数のジョブを前記複数のプロセッサに均等に割り当てたときの一つのジョブにおける演算時間の総計とが、ほぼ等しい時には、前記プロセッサのそれぞれには、前記ジョブの通信時間と演算時間との比の平均値が、前記プロセッサの数分の 1 になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 3】 請求項 1 または請求項 2 に記載の並列処理方法におけるジョブの割り当て方法において、各プロセッサには、通信時間と演算時間との比が、前記所定の値あるいは前記プロセッサの数分の 1 よりも大きいジョブと、小さいジョブとを交互に割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 4】 ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなるジョブの複数の個を、前記複数のプロセッサにより並列に処理する並列処理方法において、

前記複数のジョブには、 $N$  ( $N$  は整数) から  $M$  ( $N$  より大きい整数) まで連続する整数で番号付けを行い、直前に割り当ての済んだジョブの番号  $J$  が  $(N+M)/2$  以下であれば、番号が  $N+(M-J-1)$  のジョブを次のジョブとして割り当て、直前に割り当ての済んだジョブの番号  $J$  が  $(N+M)/2$  より大きいならば、番号が  $N+(M-J)$  のジョブを次のジョブとして割り当てるまたは、直前に割り当ての済んだジョブの番号  $J$  が  $(N+M)/2$  より小さいならば、番号が  $N+(M-J)$  のジョブを次のジョブとして割り当て、直前に割り当ての済んだジョブの番号  $J$  が  $(N+M)/2$  以上であれば、番号が  $N+(M-J-1)$  のジョブを次のジョブとして割り当てて、

前記通信時間と前記演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 5】 ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなるジョブの複数の個を、前記複数のプロセッサにより並列に処理する並列処理方法において、

前記複数のプロセッサを、グループ内の前記プロセッサの数が等しい  $G$  個 ( $G$  は 2 以上の整数) のプロセッサ・グループに分割すると共に、前記複数のジョブを、グループ内の前記ジョブの数がほぼ等しい  $G$  個のジョブ・グループに分割し、

前記プロセッサ・グループと、前記ジョブ・グループとを 1 対 1 に対応させて、一つのジョブ・グループ内のジョブを、対応する一つのプロセッサ・グループ内のプロセッサに割り当てるものであって、

前記通信時間と前記演算時間とからなるサイズと、前記通信時間と前記演算時間との比とが近似するジョブは、前記ジョブ・グループの異なるものにそれぞれ属するようにすると共に、

前記通信時間と前記演算時間とからなるサイズと、前記通信時間と演算時間との比とが近似するジョブの、前記ジョブ・グループのそれぞれ内での割り当て順番を、前記複数のジョブ・グループにおいて互いに異なるように割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 6】 請求項 5 に記載の並列処理方法におけるジョブの割り当て方法において、

前記複数のジョブには、 $N$  ( $N$  は整数) から  $M$  ( $N$  より大きい整数) まで連続する整数で番号付けを行い、 $g$  番目のジョブ・グループは、前記ジョブの番号が 0 から順に番号付けされたものに置き換えられたとき、前記  $G$  で除算した余りが  $g$  に等しい番号のジョブにより構成されるようにして、

前記通信時間と前記演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 7】 請求項 5 に記載の並列処理方法におけるジョブの割り当て方法において、

前記複数のジョブには、 $N$  ( $N$  は整数) から  $M$  ( $N$  より大きい整数) まで連続する整数で番号付けを行い、 $g$  番目のジョブ・グループは、前記ジョブの番号が 0 から順に番号付けされたものに置き換えられたとき、ジョブ番号を  $2 \times G$  で除算した余りが  $g$  に等しいジョブおよびジョブ番号を  $2 \times G$  で除算した余りが  $(2 \times G - g - 1)$  に等しいジョブにより構成されるようにして、前記通信時間と前記演算時間との比の平均値が、所定の

## 3

値になるように各ジョブを割り当てることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 8】請求項 6 または請求項 7 に記載の並列処理方法におけるジョブの割り当て方法において、  
g 番目のジョブ・グループを構成するジョブのうち、最初にプロセッサに割り当てられるジョブは、  
ジョブ番号 J をプロセッサ・グループ数 G で除算した時の商が互いに等しいジョブにより構成され、その商を用いて、0 以上、 $B = (M - N + 1) / G$  以下の整数で番号付けされるジョブ・ブロックのうち、

$(B / G) \times g$  以上、 $(B / G) \times (g + 1)$  未満の番号を有するジョブ・ブロックに含まれる番号のジョブであることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 9】請求項 6 または請求項 7 に記載の並列処理方法におけるジョブの割り当て方法において、  
対応するプロセッサ・グループを構成するプロセッサ・エレメント数が m である g 番目のジョブ・グループを構成するジョブのうち、最初にプロセッサに割り当てられるジョブは、

ジョブ番号 J をプロセッサ・グループ数 G で除算した時の商が互いに等しいジョブにより構成され、その商を用いて、0 以上、 $B = (M - N + 1) / G$  以下の整数で番号付けされるジョブ・ブロックのうち、

$(B / G / m) \times g$  以上、 $(B / G / m) \times (g + 1)$  未満の番号を有するジョブ・ブロックに含まれる番号のジョブであることを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 10】請求項 8 に記載の並列処理方法におけるジョブの割り当て方法において、

g 番目のジョブ・グループにおいて直前に割り当ての済んだジョブが b 番目のジョブ・ブロックに含まれる場合に、

$(b + 1)$  番目のジョブ・ブロックに g 番目のジョブ・グループのジョブがあれば  $(b + 1)$  番目のジョブ・ブロック内のジョブを、そうでなければ 0 番目のジョブ・ブロック内のジョブを、

または  $(b - 1)$  が 0 以上なら  $(b - 1)$  番目のジョブ・ブロック内のジョブを、そうでなければ g 番目のジョブ・グループ内でまだ割り当ての行われていない最大番号のジョブを、

前記 g 番目のジョブ・グループの次のジョブとして割り当ててことを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 11】請求項 9 に記載の並列処理方法におけるジョブの割り当て方法において、

g 番目のジョブ・グループにおいて直前に割り当ての済んだジョブが b 番目のジョブ・ブロックに含まれる場合に、

$b + (B / m)$  番目のジョブ・ブロックに g 番目のジョ

## 4

ブ・グループのジョブがあれば  $b + (B / m)$  番目のジョブ・ブロック内のジョブを、そうでなければ  $b + 1$  を  $(B / m)$  で除算した余りに等しい番号のジョブ・ブロック内のジョブを、

または  $b - (B / m)$  が 0 以上なら  $b - (B / m)$  番目のジョブ・ブロック内のジョブを、b が 0 なら  $(B / m)$  で除算した余りが  $(B / m) - 1$  に等しいジョブ・ブロック番号のうち g 番目のジョブ・グループのジョブを含む最大の番号のジョブ・ブロック内のジョブを、b がそれ以外であれば、 $(B / m)$  で除算した余りが、

$(b - 1)$  を  $(B / m)$  で除算した余りと等しいジョブ・ブロック番号のうち g 番目のジョブ・グループのジョブを含む最大の番号のジョブ・ブロック内のジョブを、前記 g 番目のジョブ・グループの次のジョブとして割り当ててことを特徴とする並列処理方法におけるジョブの割り当て方法。

【請求項 12】n 個 (n は正の整数) の縮約シェルを用いて表現される分子のエネルギーを計算する分子軌道計算を、ホスト計算機と複数のプロセッサとにより構成される計算機システムを用いて行う並列処理方法であって、

縮約シェル R, S, T, U のそれぞれに含まれる原始シェル r, s, t, u のそれぞれの成分である原始基底関数 i, j, k, l をインデックスとして用いて表わされる 2 電子積分関数 g の関数値  $g(i, j, k, l)$  と；  
前記原始基底関数 k をひとつの構成要素とする縮約基底関数 K および前記原始基底関数 l をひとつの構成要素とする縮約基底関数 L とをインデックスとして用いて表わされる密度行列 P の要素  $P(K, L)$  と；係数 A1 との積  $A1 \cdot P(K, L) \cdot g(i, j, k, l)$  の全ての縮約基底関数に関する総和  $f1(I, J)$  と、

前記 2 電子積分関数の関数値  $g(i, k, j, l)$  と；  
前記密度行列 P の要素  $P(K, L)$  と；係数 A2 との積  $A2 \cdot P(K, L) \cdot g(i, k, j, l)$  の全ての縮約基底関数に関する総和  $f2(I, J)$  との和  $f(I, J) = f1(I, J) + f2(I, J)$  の、前記原始基底関数 i, j をひとつの構成要素とする前記縮約基底関数 I, J に含まれる全ての前記原始基底関数に関する和で表わされるフォック行列 F の全ての行列要素  $F(I, J)$  の計算を、

最も外側のループは、 $R \leq n_{\max}$  ( $n_{\max}$  は n 個の縮約シェルに付与された番号の最大値) および  $T \leq R$  なる関係を満たす前記縮約シェル R と T との組み合わせに関するループとし、

前記最も外側のループの内側の 2 番目は前記縮約シェル S に関するループ、前記 2 番目よりも内側の 3 番目は前記縮約シェル U に関するループとするか、あるいは前記 2 番目は前記縮約シェル U に関するループ、前記 3 番目は前記縮約シェル S に関するループとし、

前記縮約シェル S のとりうる値の範囲を R 以下とし、

## 5

前記縮約シェルUのとりうる値の範囲をR以下とし、  
前記3番目のループの内側で、所定の2電子積分の計算  
およびその結果を用いた所定のフォック行列要素の一部  
の計算を行うものであって、

前記2番目および3番目のループをひとまとまりとして  
1つのジョブ単位を形成し、前記ジョブ単位毎に前記プ  
ロセッサに処理を割り当てる並列処理方法において、  
前記ジョブは、前記縮約シェルRの軌道量子数が高いジ  
ョブは、小さい番号を割り当て、前記縮約シェルRの軌  
道量子数が低いジョブには大きい番号を割り当てて形成  
10 することを特徴とする並列処理方法。

【請求項13】請求項12に記載の並列処理方法におい  
て、

請求項1、請求項2、請求項3、請求項5のいずれかに  
記載のジョブの割り当て方法を用いて、前記複数のプロ  
セッサへの前記ジョブの割り当てを行うことを特徴とす  
る並列処理方法。

【請求項14】請求項12に記載の並列処理方法におい  
て、

軌道量子数が高い縮約シェルには、小さい縮約シェル番  
号を割り当て、前記軌道量子数が低い縮約シェルには、  
大きい縮約シェル番号を割り当てて前記ジョブの番号  
を、 $R(R+1)/2+T$ の算出式により定めて、N  
(Nは整数)からM(Nより大きい整数)まで連続する  
整数で番号付けを行い、

請求項4、請求項6、請求項7、請求項8、請求項9、  
請求項10、請求項11のいずれかに記載のジョブの割  
り当て方法を用いて、前記複数のプロセッサへの前記ジ  
ョブの割り当てを行うことを特徴とする並列処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】この発明は、ホスト計算機  
と、このホスト計算機と共通のバスを通じて接続される  
複数のプロセッサにより構成される計算機システムを用  
いて、複数のジョブを、複数のプロセッサにより並  
列に処理する並列処理方法におけるジョブの割り当て方  
法に関する。また、大規模で特定の対称性を有する行列  
要素計算、特に非経験的分子軌道法を用いた分子シミュ  
レーションにおいてフォック行列要素計算を高速に処理  
するための並列処理方法に関する。

【0002】

【従来の技術】化学の分野において、分子の状態や挙動  
を数値的に解析する手法として、分子軌道法、分子動力  
学法、モンテカルロ法などがある。その中でも、非経験  
的分子軌道計算法は、第一原理に基づいた量子力学的計  
算で、分子中の電子の挙動を記述することを目的として  
いる。そのため、この手法は、分子シミュレーションの  
基盤として位置づけられ、物質構造や化学反応の詳細な  
解析に用いられている工業的に重要な手法である。

【0003】非経験的分子軌道計算法では、分子を構成

## 6

する原子の原子核と軌道電子との距離の2乗に経験的な  
定数を乗じたものを指数とする指数関数の逆数あるいは  
それらの線形結合を基底関数とし、この基底関数を1つ  
の原子に対して複数個用意する。これらの基底関数の線  
形結合で、分子内の軌道電子の波動関数、すなわち分子  
軌道を記述する。

【0004】分子軌道における基底関数の線形結合係数  
を決めることが、非経験的分子軌道計算法での主要な処  
理であるが、その計算には、基底関数の数の4乗に比例  
した計算量と記憶容量を必要とする。そのため、非経験  
的分子軌道計算法は、現状では、100原子程度の規模  
の分子系に適用されているに過ぎない。生命現象／化学  
現象の分子論的解明を、より現実的なものとするために  
は、数1000原子規模の分子系への適用も視野に入れ  
た、非経験的分子軌道計算専用の計算システムの開発が  
必須である。

【0005】このように規模の大きい非経験的分子軌道  
計算専用の計算システムの例として、文献1(白川他  
“超高速分子軌道計算専用機MOEのアーキテク  
20 ャ”，電子情報通信学会技術報告，vol. CPSY9  
6-46, no. 5, pp. 45-50, 1996)あ  
るいは文献2(稲畑他，“PPRAM-MOE：分子軌  
道計算専用サーバMOEのプロセッシング・ノードLS  
1”，電子情報通信学会技術報告，vol. CPSY9  
8-21, no. 4, pp. 77-84, 1998)に  
記載のシステムがある。

【0006】これらの文献に記載の計算システムは、ホ  
スト計算機と、例えば100個のプロセッサ(以下の説  
明においては、この並列処理のためのプロセッサを、プ  
ロセッサ・エレメントと呼ぶことにする。)とが、共通  
のバスにより接続された構成の並列処理システムで、非  
経験的分子軌道計算を高速に行うことを目的としてい  
る。

【0007】これらのシステムは、個々の並列プロセッ  
サ・エレメントが高い演算処理能力を有する上、システ  
ム全体を低価格で実現することが可能であるため、コス  
トパフォーマンスに優れた計算システムを提供すること  
ができる。

【0008】

【発明が解決しようとする課題】非経験的分子軌道計算  
のように、計算量が膨大なものである場合、並列処理用  
の複数のプロセッサ・エレメントを効率的に運用し  
て、より高速化することが重要である。特に、上記の文  
献に記載のシステムにおいては、各プロセッサ・エレメ  
ントは、共通のバスを通じてホスト計算機と通信を行っ  
て、必要なデータの授受を行いながら、計算処理を実行  
するので、できるだけ、各プロセッサ・エレメントにお  
ける通信の待ち時間をなくして、効率的な処理を行える  
ようにすることが、高速処理のために重要である。

【0009】しかしながら、単純に、非経験的分子軌道

## 7

計算について、ジョブを形成し、並列プロセッサ・エレメントに割り付けた場合には、ジョブ毎に、計算量（計算時間や通信量）が異なるため、通信のオーバーヘッド（待ち時間）が生じることにより、プロセッサ・エレメントの個数に比例した処理性能の向上が得られなくなる問題がある。

【0010】並列処理を行うプロセッサ・エレメントの個数が増加すると、ホスト計算機とプロセッサ・エレメントとの間で送受信されるトータルのデータ量は変わらないが、プロセッサ・エレメントによる処理能力が個数に比例して増加する。その結果、一定時間内で行わなければならない通信の量（平均通信量）が増加し、プロセッサ・エレメント数が、ある個数を超えると平均通信量が通信網の処理能力（通信性能）を超えてしまう。このことが、通信のオーバーヘッドを発生させる一因となる。

【0011】また、短い時間内に通信要求が集中的に発生し、合計の通信量が一時的に通信網の通信性能を超えてしまうような、時間的に不均一に通信負荷が発生するような場合にも、通信のオーバーヘッドが発生し、並列化効率を劣化させる原因となる。これは、平均通信量が通信網の通信性能を超えない場合であっても起こり得る。

【0012】したがって、発生するトータルの通信量を少なくすることと、通信負荷の発生を時間的に均一にすること（通信負荷の時間的な分散）が、並列計算機で効率的な処理を行うために不可欠である。

【0013】例えば、複数個のジョブのそれぞれが、計算量（計算時間；なお、この計算には、例えばROMを用いた変換処理なども含まれるものとする。この明細書では、計算と演算とは同義のものとして使用している）と通信量（通信時間）とが均等なジョブであれば、共通のバスを通じて行う通信タイミングを遅延させるように、各プロセッサ・エレメントにジョブを割り振ることにより、通信のオーバーヘッドを生じることなく、並列処理が行える。

【0014】文献3（特開平5-324581号公報）には、多体問題の計算を並列処理する場合に計算負荷を均一化する負荷分散方法が開示されている。この文献3に記載の負荷分散方法では、各ジョブのサイズが全て等しいことを前提としており、プロセッサ・エレメントに割り当てるジョブの個数を均等にすることにより、計算負荷を均等にしている。

【0015】しかし、非経験的分子軌道計算においては、各ジョブ毎に、計算ループの回数が異なり、また、ホスト計算機から取得するデータ量の違いによる通信量も異なる。このため、ジョブの個数を均等にするだけでは計算負荷を均等にすることができない問題がある。

【0016】上記の文献の他にも、文献4（特開平6-35870号公報）、文献5（特開平6-243112

## 8

号公報）、文献6（特開平7-295942号公報）、文献8（特開平8-185377号公報）、文献9（特開平10-240698号公報）などにも、複数のプロセッサ・エレメントにより並列処理を行う場合の負荷分散方法が開示されている。

【0017】しかし、これら文献4、文献5、文献6、文献7、文献9に記載の負荷分散方法では、いずれもプロセッサ・エレメントの負荷の状態を測定あるいは予測した上でジョブのスケジューリングを行うものであるもので、測定あるいは予測のための専用の機能を設けなければならない上に、測定あるいは予測のために付加的な処理が必要となるという問題があった。

【0018】また、文献8に記載の分散計算システムでは、分散システムを均等な能力を有するグループに分け、グループ単位で負荷分散を行うようにすることが開示されているが、通信負荷の分散に関しては考慮されていないため、通信のオーバーヘッドが大きくなってしまいうおそれがあるという問題がある。

【0019】この発明は、上述の点にかんがみ、ジョブ毎に計算負荷や通信負荷の大きさが異なる場合においても、専用の機構を設けずに、簡単な方法で、負荷を均等化できるジョブの割り当て方法を提供して、効率的な並列処理を可能にし、システム全体の処理時間を短縮化できるようにすることを目的とする。

【0020】

【課題を解決するための手段】上記課題を解決するために、請求項1の発明の並列処理方法におけるジョブの割り当て方法は、ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなる複数個のジョブを、前記複数個のプロセッサにより並列に処理する並列処理方法において、前記プロセッサのそれぞれには、前記ジョブの通信時間と演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする。

【0021】請求項2の発明は、請求項1に記載の並列処理方法におけるジョブの割り当て方法において、前記複数個のジョブの全体の通信時間と、前記複数個のジョブを前記複数個のプロセッサに均等に割り当てたときの一つのジョブにおける演算時間の総計とが、ほぼ等しい時には、前記プロセッサのそれぞれには、前記ジョブの通信時間と演算時間との比の平均値が、前記プロセッサの数分の1になるように各ジョブを割り当てることを特徴とする。

【0022】また、請求項3の発明は、請求項1または請求項2に記載の並列処理方法におけるジョブの割り当て方法において、各プロセッサには、通信時間と演算時間との比が、前記所定の値あるいは前記プロセッサの数分の1よりも大きいジョブと、小さいジョブとを交互に

10

20

30

40

50



割り当てることを特徴とする。

【0023】また、請求項4の発明は、ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなるジョブの複数個を、前記複数個のプロセッサにより並列に処理する並列処理方法において、前記複数個のジョブには、 $N$  ( $N$ は整数) から  $M$  ( $N$ より大きい整数) まで連続する整数で番号付けを行い、直前に割り当ての済んだジョブの番号  $J$  が  $(N+M)/2$  以下であれば、番号が  $N+(M-J-1)$  のジョブを次のジョブとして割り当て、直前に割り当ての済んだジョブの番号  $J$  が  $(N+M)/2$  より大きいならば、番号が  $N+(M-J)$  のジョブを次のジョブとして割り当てるまたは、直前に割り当ての済んだジョブの番号  $J$  が  $(N+M)/2$  より小さいならば、番号が  $N+(M-J)$  のジョブを次のジョブとして割り当て、直前に割り当ての済んだジョブの番号  $J$  が  $(N+M)/2$  以上であれば、番号が  $N+(M-J-1)$  のジョブを次のジョブとして割り当て、前記通信時間と前記演算時間との比の平均値が、所定の値になるように各ジョブを割り当てることを特徴とする。

【0024】また、請求項5の発明は、ホスト計算機と、このホスト計算機と共通のバスを通じて接続される複数のプロセッサにより構成される計算機システムを用いて、前記ホスト計算機と前記プロセッサとの間の通信処理と前記プロセッサにおける演算処理とからなるジョブの複数個を、前記複数個のプロセッサにより並列に処理する並列処理方法において、前記複数個のプロセッサを、グループ内の前記プロセッサの数がほぼ等しい  $G$  個 ( $G$ は2以上の整数) のプロセッサ・グループに分割すると共に、前記複数個のジョブを、グループ内の前記ジョブの数がほぼ等しい  $G$  個のジョブ・グループに分割し、前記プロセッサ・グループと、前記ジョブ・グループとを1対1に対応させて、一つのジョブ・グループ内のジョブを、対応する一つのプロセッサ・グループ内のプロセッサに割り当てるものであって、前記通信時間と演算時間とからなるサイズと、前記通信時間と前記演算時間との比とが近似するジョブは、前記ジョブ・グループの異なるものにそれぞれ属するようにすると共に、前記通信時間と演算時間とからなるサイズと、前記通信時間と前記演算時間との比とが近似するジョブの、前記ジョブ・グループのそれぞれ内での割り当て順番を、前記複数個のジョブ・グループにおいて互いに異なるように割り当てることを特徴とする。

【0025】また、請求項12の発明は、 $n$  個 ( $n$ は正の整数) の縮約シェルを用いて表現される分子のエネルギーを計算する分子軌道計算を、ホスト計算機と複数個のプロセッサとにより構成される計算機システムを用いて行う並列処理方法であって、縮約シェル  $R$ ,  $S$ ,  $T$ ,

$U$ のそれぞれに含まれる原始シェル  $r$ ,  $s$ ,  $t$ ,  $u$ のそれぞれの成分である原始基底関数  $i$ ,  $j$ ,  $k$ ,  $l$  をインデックスとして用いて表わされる2電子積分関数  $g$  の関数値  $g(i, j, k, l)$  と; 前記原始基底関数  $k$  をひとつの構成要素とする縮約基底関数  $K$  および前記原始基底関数  $l$  をひとつの構成要素とする縮約基底関数  $L$  とをインデックスとして用いて表わされる密度行列  $P$  の要素  $P(K, L)$  と; 係数  $A1$  との積  $A1 \cdot P(K, L) \cdot g(i, j, k, l)$  の全ての縮約基底関数に関する総和  $f1(I, J)$  と、前記2電子積分関数の関数値  $g(i, k, j, l)$  と; 前記密度行列  $P$  の要素  $P(K, L)$  と; 係数  $A2$  との積  $A2 \cdot P(K, L) \cdot g(i, k, j, l)$  の全ての縮約基底関数に関する総和  $f2(I, J)$  との和  $f(I, J) = f1(I, J) + f2(I, J)$  の、前記原始基底関数  $i$ ,  $j$  をひとつの構成要素とする前記縮約基底関数  $I$ ,  $J$  に含まれる全ての前記原始基底関数に関する和で表わされるフォック行列  $F$  の全ての行列要素  $F(I, J)$  の計算を、最も外側のループは、 $R \leq n_{\max}$  ( $n_{\max}$  は  $n$  個の縮約シェルに付与された番号の最大値) および  $T \leq R$  なる関係を満たす前記縮約シェル  $R$  と  $T$  との組み合わせに関するループとし、前記最も外側のループの内側の2番目は前記縮約シェル  $S$  に関するループ、前記2番目よりも内側の3番目は前記縮約シェル  $U$  に関するループとするか、あるいは前記2番目は前記縮約シェル  $U$  に関するループ、前記3番目は前記縮約シェル  $S$  に関するループとし、前記縮約シェル  $S$  のとりうる値の範囲を  $R$  以下とし、前記縮約シェル  $U$  のとりうる値の範囲を  $R$  以下とし、前記3番目のループの内側で、所定の2電子積分の計算およびその結果を用いた所定のフォック行列要素の一部の計算を行うものであって、前記2番目および3番目のループをひとまとまりとして1つのジョブ単位を形成し、前記ジョブ単位毎に前記プロセッサに処理を割り当てる並列処理方法において、前記ジョブは、前記縮約シェル  $R$  の軌道量子数が高いジョブには小さい番号を割り当て、前記縮約シェル  $R$  の軌道量子数の低いジョブには大きい番号を割り当てて形成することを特徴とする。

【0026】また、請求項13の発明は、請求項12に記載の並列処理方法において、請求項1、請求項2、請求項3、請求項5のいずれかに記載のジョブの割り当て方法を用いて、前記複数個のプロセッサへの前記ジョブの割り当てを行うことを特徴とする。

【0027】また、請求項14の発明は、請求項12に記載の並列処理方法において、軌道量子数が高い縮約シェルには、小さい縮約シェル番号を割り当て、前記軌道量子数が低い縮約シェルには、大きい縮約シェル番号を割り当てて前記ジョブの番号を、 $R(R+1)/2 + T$  の算出式により定めて、 $N$  ( $N$ は整数) から  $M$  ( $N$ より大きい整数) まで連続する整数で番号付けを行い、請求項4、請求項6、請求項7、請求項8、請求項9、請求

項 10、請求項 11 のいずれかに記載のジョブの割り当て方法を用いて、前記複数のプロセッサへの前記ジョブの割り当てを行うことを特徴とする。

【0028】

【作用】上述の構成の請求項 1 の発明によれば、ジョブの通信時間と演算時間との比の平均値が、所定の値になるように、複数のプロセッサにジョブが割り当てられるので、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【0029】また、請求項 2 の発明によれば、複数のジョブの全体の通信時間と、複数のジョブを複数のプロセッサに均等に割り当てたときの一つのジョブにおける演算時間の総計とが、ほぼ等しい場合に、ジョブの通信時間と演算時間との比の平均値が、プロセッサの数分の 1 になるようにされるので、各プロセッサにおいて通信も演算も行っていない時間帯が生じることがなく、並列の複数のプロセッサのそれぞれで前記演算時間が終了した時には、処理が終了となる。したがって、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【0030】請求項 3 の発明によれば、通信時間と演算時間との比が、所定の値あるいはプロセッサの数分の 1 よりも大きいものと、小さいものとを交互に割り当てるという簡単な方法により、通信処理と演算処理とを効率的に行うことができるようになる。

【0031】請求項 4 の発明によれば、ジョブには、通信時間と演算時間との比の小さいものから順に、あるいは大きいものから順に、 $N$  ( $N$  は整数) から  $M$  ( $N$  より大きい整数) まで連続する整数で番号付けが行われている。そして、複数のジョブのサイズは、ばらつきが小さいものとされている。

【0032】そして、プロセッサにおける前処理のジョブ番号が、全体のジョブ番号のなかの中間値よりも大きいときには、次のジョブとしては、ジョブ番号が小さいものが割り当てられ、前処理のジョブ番号が、全体のジョブ番号のなかの中間値よりも小さいときには、次のジョブとしては、ジョブ番号が大きいものが割り当てられる。

【0033】したがって、ジョブの通信時間と演算時間との比の平均値が、ほぼプロセッサの数分の 1 になるようにされ、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【0034】また、請求項 5 の発明においては、複数のプロセッサは、グループ内のプロセッサの数がほぼ等しい  $G$  個 ( $G$  は 2 以上の整数) のプロセッサ・グループに分割される。また、複数のジョブは、グループ内のジョブの数が、プロセッサ・グループの数に等しい  $G$  個のジョブ・グループに分割される。そして、プロセッサ・グループと、ジョブ・グループとが 1 対 1 に対応させられて、一つのジョブ・グループ内のジョブが、対応す

る一つのプロセッサ・グループ内のプロセッサに割り当てられる。

【0035】この場合に、通信時間と演算時間とからなるサイズと、通信時間および演算時間が近似するジョブは、ジョブ・グループの異なるものにそれぞれ属するように分散され、しかも、通信時間と演算時間とからなるサイズと、通信時間および演算時間が近似するジョブの、ジョブ・グループのそれぞれ内での割り当て順番が、複数のジョブ・グループにおいて互いに異なるように割り当てられて分散され、複数のジョブによる通信処理が共通バス上で輻輳状態にならないようにされる。

【0036】したがって、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【0037】また、請求項 12 の発明は、非経験的分子軌道計算におけるフォック行列計算アルゴリズムによるフォック行列の計算を、ホスト計算機とプロセッサ・エレメントとの間の通信性能およびホスト計算機の処理性能に律速されることなく、高速に行うことができる並列処理方法である。

【0038】この場合、縮約シェル  $R$  および  $T$  に関するループはホスト計算機で制御され、 $R$  と  $T$  の組み合わせの固定された  $S$  および  $U$  に関するループの部分がプロセッサ・エレメントに割り当てられる。

【0039】 $R \geq T$ 、 $S$ 、 $U$  の関係があるので、 $R$  の番号の大きいものは、ループ回数も大きい。一方、 $R$  の番号が小さいものは、ループ回数が少ない。また、軌道量子数が高いシェルは、計算に必要な密度行列が多数必要になり、その分、通信量が大きくなり、軌道量子数が低いシェルは、密度行列が少なく、その分、通信量が少なくなる。

【0040】このため、この請求項 12 のように、縮約シェル  $R$  の軌道量子数が高いジョブには小さいジョブ番号が割り当てられ、縮約シェル  $R$  の軌道量子数が低いジョブには大きいジョブ番号が割り当てられると、演算処理と通信処理との和からなるジョブのサイズのばらつきが小さくなる。

【0041】したがって、並列処理の複数のプロセッサ・エレメントへのジョブの、効率的な割り当てを容易に行えるようになる。

【0042】そこで、請求項 13 の発明のように、請求項 12 の分子軌道計算の並列処理方法において、請求項 1、請求項 2、請求項 3、請求項 5 のいずれかに記載のジョブの割り当て方法を用いて、前記複数のプロセッサへの前記ジョブの割り当てを行うことにより、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【0043】また、請求項 14 の発明の場合には、軌道量子数の高い縮約シェルには、小さい縮約シェルの番号を割り当て、軌道量子数の低い縮約シェルには、大きい

10

20

30

40

50

縮約シェルの番号を割り当て、 $R(R+1)/2+T$ の算出式により定めて、 $N$  ( $N$ は整数) から  $M$  ( $N$ より大きい整数) まで連続する整数で番号付けを行うので、結果として得られるジョブ番号は、通信時間と演算時間の比の大きさに概略したがった順番となる。

【0044】このため、請求項12の分子軌道計算の並列処理方法において、請求項4、請求項6、請求項7、請求項8、請求項9、請求項10、請求項11のいずれかに記載のジョブの割り当て方法を用いて、前記複数のプロセッサへの前記ジョブの割り当てを行うことにより、通信処理も演算処理も効率的に行われ、処理終了までの時間を短縮化することができる。

【0045】

【発明の実施の形態】以下、この発明による並列処理方法およびそのジョブの割り当て方法を、非経験的分子軌道計算法におけるフォック行列の計算アルゴリズムを行う場合に適用した実施の形態について、図を参照しながら説明する。

【0046】この発明による並列処理方法およびそのジョブの割り当て方法の実施の形態を説明する前に、非経験的分子軌道法、それに用いる計算機システムの構成例および、その非経験的分子軌道法におけるフォック行列の新規な計算アルゴリズムについて説明する。

【0047】〔非経験的分子軌道法の概要〕非経験的分子軌道計算法では、分子の状態を表わす波動関数 $\Psi$ を、分子中の電子の空間的な軌道に相当する電子軌道関数 $\phi_\mu$ を用いて記述する。ここで $\mu$ は複数ある分子軌道の $\mu$ 番目という意味の添え字である。分子軌道 $\phi_\mu$ は、原子軌道 $\chi_I$ の線形結合で、図33の(数式1)のように近似的に表わされる。

【0048】ここで、(数式1)において、 $I$ は複数ある原子軌道の $I$ 番目という意味の添え字である。なお、原子軌道は基底関数とも呼ばれることがある。この明細書中では、以降、原子軌道のことを基底関数と呼ぶ。また、(数式1)に現れる $C_{I\mu}$ は、線形結合係数である。(数式1)における $I$ に関する総和は、計算の対象とする分子を構成する全ての基底関数に関するものである。

【0049】さて、量子力学的に分子軌道を記述するためには、良く知られるパウリの排他律を、分子内の電子の状態が満たさなければならない。電子のスピンを考慮に入れて、パウリの排他律を満たすように、 $2n$ 電子系の分子の状態 $\Psi$ を記述する表式として、図33の(数式2)のようなスレーター行列式が用いられる。ここで、(数式2)において、 $\alpha(x)$ および $\beta(x)$ 、 $x$ 番目の電子のスピンが、それぞれ、上向きおよび下向きの状態を表わしている。

【0050】 $2n$ 電子系に対するハミルトニアン $H$ は、1電子部分 $H_1$ と2電子部分 $H_2$ との和という形式で、図33の(数式3)から(数式5)のように書き表され

る。

【0051】図23の(数式4)において、右辺の $(\cdot \cdot \cdot)$ 内の第1項は、電子 $p$ の運動エネルギー、第2項は $p$ 番目の電子と $A$ 番目の原子核との相互作用である。

(数式4)において、 $\Sigma_p$  (この明細書で $\Sigma_i$ は、 $i$ についての総和を取ることを表すものとする。以下、同じ)は全電子に関する総和、 $\Sigma_A$ は全原子核に関する総和、 $Z_A$ は原子核 $A$ の電荷、 $r_{pA}$ は電子 $p$ と原子核 $A$ との距離である。

10 【0052】また、(数式5)は、電子 $p$ と電子 $q$ との間の相互作用を表わしており、 $\Sigma_p \Sigma_q (>p)$ は2個の電子の組み合わせに関する総和、 $r_{pq}$ は電子 $p$ 、 $q$ 間の距離である。

【0053】上記のハミルトニアン $H$ と、(数式2)のスレーター行列式とを用いると、分子エネルギーの期待値 $\epsilon$ が、図34の(数式6)～(数式9)のように表わされる。

20 【0054】(数式6)において、 $\Sigma_\mu$ および $\Sigma_\nu$ は、 $n$ 個( $n$ は正の整数)ある分子軌道に関する総和である。(数式7)は「コア積分」と呼ばれ、代表として番号1の電子について書かれている。また、(数式8)および(数式9)は、それぞれ「クーロン積分」および「交換積分」と呼ばれ、代表として電子1および電子2について書かれている。なお、(数式6)から(数式9)において、 $d\tau$ 、 $d\tau_1$ 、 $d\tau_2$ は、位置座標空間での体積積分を表わす。

30 【0055】(数式6)を基底関数を用いて書き直すと、図35に示す(数式10)～(数式13)に示すようなものになる。(数式13)で表わされる積分を、「2電子間反発積分」あるいは省略して「2電子積分」と呼ぶ。

【0056】(数式10)で表わされる分子エネルギーの期待値 $\epsilon$ は、 $C_{I\mu}$ という未知数を含んでおり、このままでは数値が得られない。 $C_{I\mu}$ は、(数式1)における線形結合定数であり、 $\mu$ は1から $n$ (分子軌道の数)の整数、 $I$ は1から $N$ ( $N$ が基底関数の数であり、正の整数)の整数である。以下では、 $C_{I\mu}$ を要素とする $N \times n$ 行列 $C$ を係数行列と呼ぶ。

40 【0057】期待値 $\epsilon$ が最小となるように係数行列を決定し、基底状態の波動関数 $\Psi$ を求める手法の1つとして、ハートリー・フォック・ローサンの変分法(以下、HFR法と略称する)が用いられる。導出過程は省略し、HFR法の結果として得られる式を、図36の(数式14)～(数式18)に示す。

50 【0058】 $F_{IJ}$ はフォック行列要素、 $P_{KL}$ は密度行列要素と、それぞれ呼ばれる。以下の説明では、これらを $F(I, J)$ 、 $P(K, L)$ のように表記する場合がある。これらは、1から $N$ の値をとる各 $I, J, K, L$ に対して数値を持っており、それぞれ $N \times N$ 行列の形で表わされる。

【0059】(数式14)を解くことにより、係数行列が求まる。(数式14)は、1からnの間の全ての $\mu$ 、および1からNの間の全てのIに対して存在するので、 $n \times N$ 本の連立方程式になっている。

【0060】(数式14)を解いて得られた係数行列Cの計算には、密度行列Pが用いられている。密度行列Pは、(数式18)に示すように係数行列Cから計算される。そのため、具体的な計算手順としては、まず、適当に係数行列Cを与えておき、それを用いて計算した密度行列Pを使って、(数式15)でフォック行列Fを計算し、(数式14)の連立方程式を解いて新たな係数行列Cを得る。密度行列Pの元となるCと、結果として得られるCとの間の差が十分小さく、すなわち自己無撞着になるまで、上記の計算を繰り返し行う。この反復計算を自己無撞着計算(以下、SCF計算と称する)と呼ぶ。

【0061】実際の計算で最も時間を要するのは、(数式15)のフォック行列要素 $F_{IJ}$ の計算である。これは、全てのI、Jに対して、この(数式15)を計算しなければならないこと、および各I、Jの組み合わせに対して、密度行列要素 $P_{KL}$ のK、Lに関する和を計算しなければならないことに起因する。

【0062】SCF計算の手法には2通りある。1つはディスクストレージSCF法と呼ばれる手法で、1回目のSCF計算の際に得た2電子積分の値を全てディスクに保存しておき、2回目以降は必要な2電子積分をディスクから取り出して用いる手法である。もう1つはダイレクトSCF法と呼ばれる手法で、SCF計算の度に2電子積分の計算をやり直す手法である。

【0063】現在では、ディスク容量の制限やアクセス時間の大きさなどから、後者のダイレクトSCF法を用いるのが主流である。このダイレクトSCF法による分子軌道計算においては、SCF計算の1回あたりに、 $N^4$ にほぼ比例する個数の2電子積分の計算を行わなければならないため、2電子積分計算を高速に行うことが分子軌道計算を高速化することに直結する。

【0064】2電子積分 $G(I, J, K, L)$ 、密度行列 $P(K, L)$ 、およびフォック行列 $F(I, J)$ の対称性に関して、ここで言及しておく。

【0065】2電子積分は、(数式13)から明らかに、図26の(数式19)に示すような対称性を有している。したがって、(数式19)の内の1つに関して数値を得ることができれば、他の7つについても数値が得られたことになる。

【0066】また、図36の(数式18)から、 $P(K, L) = P(L, K)$

であることがわかり、図26の(数式15)および図25の(数式11)から、

$F(I, J) = F(J, I)$

であることがわかる。

【0067】[縮約基底関数と原始基底関数] 非経験的

分子軌道法では、図37の(数式20)に示すような基底関数が一般的に用いられる。この(数式20)において、 $r$ 、 $n$ 、 $R$ はベクトルであり、添え字 $x$ 、 $y$ 、 $z$ の付いたものがその成分である。 $r$ は電子の座標、 $n$ は電子の角運動量、 $R$ は原子核の座標である。

【0068】 $n_x + n_y + n_z = \lambda$ は、角運動量の大きさであり、軌道量子数とも呼ばれる。この軌道量子数 $\lambda$ が、0の場合にその軌道をs軌道、1の場合にその軌道をp軌道、2の場合にその軌道をd軌道などと呼ぶ。

【0069】 $\zeta_m$ は軌道指数であり、軌道の空間的な広がり具合を示す。軌道指数の異なる複数の軌道の線形結合で1つの基底関数を表わす場合があり、そのようにして表わした基底関数を縮約基底関数と呼び、線形結合係数 $d_m$ を縮約係数と呼ぶ。これに対して、線形結合される前の、図27の(数式21)の形の関数 $\psi$ を原始基底関数と呼ぶ。

【0070】縮約基底関数 $\chi$ は、I、J、K、Lのように大文字で番号付けをし、また、原始基底関数 $\psi$ は、i、j、k、lのように小文字で番号付けするのが慣例であり、本明細書中でもこれに従う。

【0071】[縮約シェルと原始シェル] 軌道量子数が1の場合の縮約基底関数には、 $n = (1, 0, 0)$ の場合、 $n = (0, 1, 0)$ の場合、 $n = (0, 0, 1)$ の場合の3通りが存在する。同様に、軌道量子数が2の場合には6通り(あるいは、基底関数の構成の仕方によっては5通り)の縮約基底関数が存在する。

【0072】(数式20)のうちの図37の(数式22)で示す部分が共通な、これら複数の縮約基底関数の集合を、縮約シェルと呼ぶ。p軌道の縮約シェルは3つの縮約基底関数で構成され、また、d軌道の縮約シェルは6つ(または5つ)の縮約基底関数で構成される。s軌道の場合にも、便宜上1つの縮約基底関数の集合を縮約シェルと呼ぶ。

【0073】(数式21)のうちの $\exp[-\zeta(r-R)^2]$ の部分が共通な、原始基底関数の集合を、同様に原始シェルと呼ぶ。縮約シェルは、R、S、T、Uのように大文字で番号付けをし、原始シェルは、r、s、t、uのように小文字で番号付けするのが慣例であり、本明細書中でもこれに従う。

【0074】分子軌道計算の実施に際しては、計算の対象とする分子を構成する原子毎に軌道量子数の異なる複数の縮約シェルを用意し、それら全ての集合を基底関数のセットとして用いる。原子核座標Rと軌道量子数 $\lambda$ との組み合わせ $(R, \lambda)$ で、1つの縮約シェルを表わすことができる。

【0075】[2電子積分の表式] 縮約基底関数で表わされる2電子積分 $G(I, J, K, L)$ は、原始基底関数を用いると、図37の(数式23)のように表わされる。ここで、 $g(i, j, k, l)$ は、図37の(数式24)のように表すことができる。

【0076】 $G(I, J, K, L)$  を、縮約基底関数で表現した2電子積分と呼び、 $g(i, j, k, l)$  を、原始基底関数で表現した2電子積分と呼ぶが、以降の説明では、どちらも単に2電子積分と呼ぶ場合がある。 $g(i, j, k, l)$  も、図37の(数式25)で示すような対称性を有している。

【0077】さて、原始基底関数 $\psi$ は、その角運動量 $n$ 、軌道指数 $\zeta$ 、原子核座標 $R$ の組み合わせで、一意的に示すことができる。 $i, j, k, l$  番目の原始基底関数が、図32に示す表1のような角運動量、軌道指数、原子核座標を有するものと仮定する。

【0078】説明の煩雑さを避けるために、以下の説明では、原始基底関数の番号 $i, j, k, l$ の代わりに、それぞれの角運動量 $a, b, c, d$ を用いて2電子積分を $[ab, cd]$ のように表わすことにする。

【0079】上記のように用意された基底関数セットを用いて2電子積分を計算する効率的な手法を、文献1 (S. Obara and A. Saika, JCP, vol. 84, no. 7, p. 3964, 1986) に従って説明する。

【0080】まず、 $a, b, c, d$ が全て $s$ 軌道、すなわち $a=0_a=(0, 0, 0)$ ,  $b=0_b=(0, 0, 0)$ ,  $c=0_c=(0, 0, 0)$ ,  $d=0_d=(0, 0, 0)$ である場合には、(数式24)の2電子積分は、図38の(数式26)～(数式34)に示すように求まる。

【0081】ここで、(数式26)に現れる $[\cdot \cdot, \cdot \cdot]^{(m)}$ は補助積分、 $m$ は補助インデックスであるが、これらについては後で述べる。(数式27)の積分範囲は、0から1である。

【0082】また、 $a, b, c, d$ のうち1つでも $s$ 軌道以外のものがある場合には、図39の(数式35)および(数式36)に示す漸化式を用いて計算する。

【0083】(数式35)で、添え字の $i$ は、 $x$ または $y$ または $z$ 成分であることを示す。また、 $1_i$ は、 $i$ 成分のみ1で、他は0であるようなベクトルである。さらに、 $N_i(n)$ は、角運動量 $n$ の $i$ 成分の値を示すものである。(数式35)は、左辺の補助積分に現れる角運動量の1つは右辺では確実に1以上減少し、また、左辺の補助積分の補助インデックスは右辺では同じあるいは1だけ増加する、という性質を有している。

【0084】補助積分 $[\cdot \cdot, \cdot \cdot]^{(m)}$ は、補助インデックス $m$ が0であるときに、2電子積分 $[\cdot \cdot, \cdot \cdot]$ に正確に一致するものであり、2電子積分の計算を補助するものである。どんなに角運動量の大きな基底関数を含んだ2電子積分であっても、(数式35)を繰り返し用いて角運動量を減少させ、最後には、全て角運動量が $(0, 0, 0)$ であるような補助積分に行き着くことができる。角運動量が $(0, 0, 0)$ であるような補助積分は、(数式26)を用いて計算できるので、その

数値と適当な係数を乗じ、加算すれば2電子積分の数値が得られる。

【0085】実際の計算は、以下のように行う。まず、(数式35)に従って、2電子積分を8つ以下の補助積分を用いた形式に表わす。ここで現れた補助積分に対して、さらに、(数式35)を適用する。このような手続きを繰り返して、全て角運動量が $(0, 0, 0)$ であるような補助積分に行き着くまでの道筋を、計算手順として記録しておく。

10 【0086】次に、(数式26)を用いて角運動量が $(0, 0, 0)$ であるような補助積分を計算し、そこから出発して、先ほどの計算手順をたどりながら補助積分の数値を計算していき、最後に目的とする2電子積分の数値を得る。

【0087】(数式35)が有するもう一つの重要な性質は、2電子積分に現れる4つの角運動量の組み合わせが同じであれば、軌道指数や原子核座標の組み合わせが異なっても、上記の計算手順としては全く同じものを用いることができることである。計算の実行に際しては、軌道指数や原子核座標に応じて補助積分に乗じる係数を変えてやるだけで良い。

20 【0088】[カットオフ] 上述したように、計算しなければならない縮約基底関数で表わした2電子積分の数は、縮約基底関数の数 $N$ に対して $N^4$ となる。実際に数値を得なければならないのは、原始基底関数で表わした2電子積分の方であるが、その総数は、縮約基底関数で表わした2電子積分の数の数倍から数10倍(縮約基底関数を構成する原始基底関数の数、すなわち縮約数に依存する)に及ぶ。

30 【0089】この個数を減らす手法として、第1に考えられるのは、(数式19)あるいは(数式25)に記した対称性を利用することである。しかしながら、この方法では最も効率化を行っても2電子積分の計算量は $1/8$ にしかならない。

【0090】もう1つの手法は、計算精度の観点から、不必要と判断できる2電子積分の計算を、積極的に排除する方法である。不必要な2電子積分の判断は、以下のように行うことができる。

40 【0091】上述したように、全ての2電子積分の数値は、(数式26)に示した全て角運動量が $(0, 0, 0)$ であるような補助積分 $[00, 00]^{(m)}$ の数値に基づいて計算される。したがって、2電子積分の数値の、フォック行列要素の数値への寄与が計算誤差の程度であるかどうかを、 $[00, 00]^{(m)}$ の数値で判断することが可能である。さらに、 $[00, 00]^{(m)}$ の数値の大きさは、(数式29)に示した関数 $K(\zeta, \zeta', R, R')$ の値から、さらに、それは、図39の(数式37)の大きさから判断することができる。

50 【0092】したがって、 $(\zeta_a, A, \zeta_b, B)$ の数値の組み合わせで、(数式26)の1つ目の関数 $K$ の大

きさを見積ること、2電子積分 $[ab, **]$ を計算する必要があるかどうかを判断し、また、 $(\zeta_c, C, \zeta_d, D)$ の数値の組み合わせで、(数式26)の2つ目の関数 $K$ の大きさを見積ること、2電子積分 $[***, cd]$ を計算する必要があるかどうかを判断することができる。

【0093】このようにして、不必要な2電子積分の計算を排除することを、「カットオフする」と呼ぶことにする。上記の例で、 $a$ および $b$ の情報だけから判断してカットオフする場合には、 $ab$ でのカットオフ、 $c$ および $d$ の情報だけから判断してカットオフする場合には、 $cd$ でのカットオフ、と呼ぶ場合がある。このように、 $ab$ だけで、あるいは $cd$ だけでカットオフするかどうかの判断ができるのは、図29の(数式37)の最大値が1で下限値が0だからである。このようにカットオフを行うことにより、計算しなければならない2電子積分は、概略で $N^2$ に比例する個数となり、計算量を大幅に低減できる。

【0094】上述のことから、 $N$ が大きい場合には、2電子積分の対称性を利用することによる効果よりも、カットオフによる計算量低減の効果の方が桁違いに大きく、これを取り入れることによって、非経験的分子軌道計算におけるフォック行列の計算に要する処理時間が大きく短縮できることがわかる。

【0095】[分子軌道計算機システムの例] 図1は、この実施の形態の方法が適用される並列計算システムの一例の全体の構成を示すもので、これは、先に説明した文献1に示されているものである。すなわち、このシステムは、ホスト計算機1に対して、共通のバス3を通じて、複数のプロセッサ・エレメント2が接続されている。

【0096】また、図2は、この実施の形態の方法が適用される並列計算システムの一例の全体の構成を示すもので、これは、先に説明した文献2に示されているものである。この図2の例のシステムでは、ホスト計算機11に対して、シリアルバス13により、ディジーチェーン型に複数のボード12が接続されている。バス13としては、例えばIEEE1394シリアルバスが用いられる。

【0097】各ボード12は、互いに数珠つなぎとされた複数のプロセッサ・エレメント14を備え、共に、バス13とこれらの複数のプロセッサ・エレメント12との間の橋渡しを行うブリッジ15を備えている。

【0098】ところで、図1あるいは図2に示した、文献1あるいは文献2のシステムでは、個々の並列プロセッサ・エレメント2あるいは14に接続できるメモリ容量に制限がある。現状では、各プロセッサ・エレメント2あるいは14に、せいぜい数10メガ・ビット(以下、Mbと記す)程度の容量のメモリを接続するのが精

一杯である。

【0099】また、ホスト計算機1あるいは11と、各プロセッサ・エレメント2あるいは14とを接続する通信網として、上述のように、比較的汎用性が高く安価なIEEE1394シリアルバスなどを用いた場合、その通信性能は、現状では、ピーク性能で、高々、400メガ・ビット/秒(以下、Mbpsと記す)程度である。

【0100】この発明の実施の形態では、このような構成の計算機システムで、高い通信性能を必要とせず、なおかつ各プロセッシング・ノードすなわちプロセッサ・エレメントに接続されているメモリが、上記のような現実的な容量であっても、効率的に計算を行なえるような新規なフォック行列生成アルゴリズムを用いる。

【0101】[新規なフォック行列生成アルゴリズムの説明] 効率的な並列計算を可能とするフォック行列生成アルゴリズムを、プログラム・コードの形式で示した図3を用いて説明する。

【0102】説明の便宜上、図3は、縮約シェル $R$ 、 $S$ 、 $T$ 、 $U$ に関する4重ループと、その内部の縮約基底 $I$ 、 $J$ 、 $K$ 、 $L$ に関する4重ループにより構成されるように記しているが、実際のアルゴリズムでは、最も外側のループは $R \leq N_{max}$  ( $N_{max}$ は $N$ 個の縮約シェルに付与された番号の最大値である。0から $N$ までの番号を付与した場合には、 $N_{max} = N$ である。) および $R \geq T$ なる関係を満たす縮約シェル $R$ と $T$ との組み合わせ( $RT$ )に関するループとする。そして、2番目は縮約シェル $S$ に関するループとし、3番目は縮約シェル $U$ に関するループとする。3番目のループの内側で、関数値 $G(R, S, T, U)$ の計算およびその結果を用いた所定のフォック行列要素 $F$ の一部の計算を行うようにする。

【0103】なお、2番目は $U$ に関するループ、3番目は $S$ に関するループとするようにしてもよい。すなわち、図3では、 $U$ に関するループが、 $S$ に関するループの内側に形成されているが、このループの順序は逆でも良い。

【0104】縮約シェル $R$ と縮約シェル $T$ との組み合わせに関するループは、ホスト計算機上で制御され、縮約シェル $R$ と縮約シェル $T$ の組み合わせの固定された縮約シェル $S$ および縮約シェル $U$ に関するループの部分の処理がプロセッサ・エレメントに割り当てられる。 $R$ と $T$ とが固定された2番目および3番目の2つのループをひとまとまりとして一つのジョブ単位を形成し、このジョブ単位毎に複数のプロセッサ・エレメント2あるいは14に処理させるようにする。

【0105】このように、複数のプロセッサ・エレメントへのジョブ割り当てが、 $RT$ ペア単位で行われることから、以下の説明においては、このアルゴリズムを、 $RT$ 並列アルゴリズムと呼ぶことにする。

【0106】このとき、ホスト計算機1あるいは11

は、全ての行列要素Pの初期値の計算と、プロセッサ・エレメントと共に行うSCF計算と、このSCF計算を継続するかどうかの決定などを行う。SCF計算に当たって、ホスト計算機1あるいは11は、複数のプロセッサ・エレメント2あるいは14に対するRとTとが固定されたジョブ単位の割り当て決定と、プロセッサ・エレメントに対して送信すべき行列Pの一部の行列要素の選択と、選択された行列要素のプロセッサ・エレメントに対する送信と、プロセッサ・エレメントから送信されてきた行列Fの一部の行列要素の受信と、行列Fを用いた行列Pの更新とを行う。

【0107】一方、プロセッサ・エレメント2あるいは14は、ホスト計算機1あるいは11から送信された行列Pの一部の行列要素の受信と、縮約シェルSに関するループの制御および縮約シェルUに関するループの制御と、関数 $G(R, S, T, U)$ または関数 $g(i, j, k, l)$ の計算と、行列Fの一部の行列要素の計算と、行列Fの一部の行列要素のホスト計算機1あるいは11に対する送信とを行う。

【0108】縮約シェルRは0から $N_{shell}-1$ まで、縮約シェルS, T, Uは0からRまでの値をとる。そして、IはRに属する全ての縮約基底番号を、JはSに属する全ての縮約基底番号を、KはTに属する全ての縮約基底番号を、LはUに属する全ての縮約基底番号を、それぞれとる。

【0109】なお、図3では、同じ縮約シェルに属するものが連続した番号となるように縮約基底が番号付けされていることを仮定して、I, J, K, Lの範囲を、それぞれ $b\_basis(R)$ から $e\_basis(R)$ 、 $b\_basis(S)$ から $e\_basis(S)$ 、 $b\_basis(T)$ から $e\_basis(T)$ 、 $b\_basis(U)$ から $e\_basis(U)$ としている。

【0110】ここで、 $b\_basis(X)$ 、 $e\_basis(X)$ は、縮約シェルXに属する縮約基底の番号の最小値および最大値である。縮約基底に関する4重ループの内部では、2電子積分 $G(I, J, K, L)$ の計算、およびフォック行列要素の計算を行なう。ここで用いられる密度行列要素は、 $P(K, L)$ 、 $P(I, J)$ 、 $P(K, J)$ 、 $P(I, L)$ の4つであり、また、計算されるフォック行列要素は、 $F(I, J)$ 、 $F(K, L)$ 、 $F(I, L)$ 、 $F(K, J)$ の4つである。

【0111】ここで、IおよびKは、それぞれ縮約シェルRおよびTを形成する縮約基底であるので、ジョブが割り当てられた時点で、これらは数個（縮約シェルがs軌道の場合には1個、p軌道の場合には3個、d軌道の場合には6個）に固定される。JおよびLは任意である。

【0112】図3では、コードを簡潔に記述するため

に、SおよびUが、0からRまでのR+1個の全ての値をとるような書き方をしているが、実際にSおよびUがその範囲の全ての値をとるわけではない。

【0113】Rに属する全ての縮約基底と、Sに属する全ての縮約基底との間で、abでのカットオフが成り立つ場合には、全ての $G(I, J, *, *)$ の計算を省略できるので、Sはそのような値をとる必要がない。また、それに伴い、 $P(I, J)$ 、 $P(K, J)$ は用いられることがなく、 $F(I, J)$ 、 $F(K, J)$ は計算されることがないので、これらをホスト計算機とプロセッサ・エレメントとの間で通信する必要がない。

【0114】同様に、Tに属する全ての縮約基底と、Uに属する全ての縮約基底との間で、cdでのカットオフが成り立つ場合には、全ての $G(*, *, K, L)$ の計算を省略できるので、Uはそのような値をとる必要がない。また、それに伴い、 $P(I, L)$ 、 $P(K, L)$ は用いられることがなく、 $F(I, L)$ 、 $F(K, L)$ は計算されることがないので、これらをホスト計算機とプロセッサ・エレメントとの間で通信する必要がない。

【0115】したがって、SおよびUに関するループを回る回数を削減して計算量を削減できるだけでなく、不必要な行列要素の通信をなくして通信量を削減することができる。

【0116】RT並列アルゴリズムを用いた処理の流れを、図4に示すフローチャートを用いて説明する。以下の説明では、図2のシステムを用いた場合とする。また、図4においては、ホスト計算機11とプロセッサ・エレメント14とは異なる処理を行うため、それぞれについてのフローチャートを併記した。

【0117】また、プロセッサエレメント14は、並列に複数（典型的には100個）がホスト計算機11に接続されていることを前提としているが、それらは同様のフローチャートに従って動作し、また、それらの間のデータの相関はないので、代表として1つのプロセッサエレメント14における処理フローを表記した。なお、図4において、破線で示した矢印の部分は、その後の処理が、その前の処理で律速されるのではなく、他の処理系からの情報の入力待ちとなることを示している。

【0118】まず、ホスト計算機11の処理手順を説明する。ホスト計算機11は、係数行列の初期設定（ステップS101）を行った後、SCFループに入る。SCFループは、新たに計算された係数行列が自己無撞着となるまで継続する。

【0119】すなわち、SCFループ内では、まず、（数式18）に従って係数行列から密度行列を計算する（ステップS102）。そして、RTループに入り、まず、プロセッサ・エレメントに割り当てるRTペア番号、すなわち2つの縮約シェルRとTの組み合わせを一意的に表わす番号を決める（ステップS103）。この例の場合には、RTペア番号は、図39の（数式38）

に示すような算出式により、定められる。

【0120】図3の記述に従うと、RTペア番号は、小さい番号から順に割り当てられることになるが、必ずしも、上述のようにする必要はなく、全てのR、Tの組み合わせに対して処理が行なわれるようにすれば、どのような順序で処理を行なうように番号付けあるいは処理の順序付けをしても良い。

【0121】次に、上記のRTペア番号に対応したジョブで必要となる密度行列情報を、そのRTペア番号が割り当てられたプロセッサ・エレメントへ送信する（ステップ104）。このとき、abおよびcdでのカットオフを考慮して、プロセッサ・エレメントにおける計算で用いられることのない密度行列データは一切送信しない。

【0122】プロセッサエレメント14は、このホスト計算機11からの密度行列情報を受信し、受信バッファメモリに格納する（ステップS201）。そして、割り当てられた（RTペア）のジョブについて、縮約シェルS、Uに関するループを制御して、フォック行列要素の計算を行う（ステップS202）。そして、計算が終了すると、求めたフォック行列情報を送信バッファメモリに格納し、その送信バッファメモリからホスト計算機11に、その求めたフォック行列情報を送信する（ステップS203）。

【0123】ホスト計算機11は、以上のようにして、あるプロセッサ・エレメント14での処理が終了して、そのプロセッサ・エレメント14からフォック行列情報を受信すると（ステップS105）、全てのプロセッサ・エレメントに対するRTペア番号の割り当てと密度行列情報の送信が終了したかどうか判断し（ステップS107）、終了していなければ、ステップS103に戻って、新たなRTペア番号の割り当てを決め、そのRTペア番号に対応する密度行列情報を、そのプロセッサ・エレメント14に送信する。

【0124】同様の操作を繰り返して、ホスト計算機11は、全てのプロセッサエレメントに対するRTペア番号の割り当てと密度行列情報の送信を行い、プロセッサエレメントからのフォック行列の受信を待つ。

【0125】ステップS106で、全てのRTペア番号の処理が終了したと判断すると、ホスト計算機11は、収集したフォック行列情報をもとに、図26の（数式14）を解き、新たな係数行列を計算する（ステップS107）。そして、新たな係数行列とその直前の係数行列とを比較して、自己無撞着な解が得られたかどうか判断する（ステップS108）。そして、両者が十分良く一致していれば、自己無撞着な解が得られたと判断し、計算を終了する。そうでなければ、ステップS108からステップS102に戻り、（数式18）に従って新たな密度行列を生成し、同様の操作を繰り返す。

【0126】なお、通信処理を計算処理でなるべく隠蔽

するために、頻繁に用いられるデータのダブル・バッファリングが、RT並列アルゴリズムにおいても適用可能である。

【0127】図5に示すように、各プロセッサ・エレメントが有するメモリのうち、密度行列情報およびフォック行列情報を格納するための領域を、AおよびBのように、2つに分けておく。

【0128】そして、密度行列情報格納領域Aに格納された密度行列要素を用いて計算（以下、この計算をタイプAの計算と呼ぶ）されたフォック行列要素は、フォック行列情報格納領域Aに、密度行列情報格納領域Bに格納された密度行列要素を用いて計算（以下、この計算をタイプBの計算と呼ぶ）されたフォック行列要素は、フォック行列情報格納領域Bに、それぞれ格納すると定める。

【0129】1つのRTペア番号に対応したジョブを、タイプAまたはタイプBの計算のどちらかに対応させ、タイプAの計算ジョブの終了後にはタイプBの計算を、タイプBの計算ジョブの終了後にはタイプAの計算を、それぞれ開始する。

【0130】タイプAの計算時には、その直前に行われたタイプBの計算ジョブによる計算結果がフォック行列情報格納領域Bに保持されており、また、密度行列情報格納領域Bは自由に上書き可能な領域となっている。

【0131】したがって、タイプAの計算ジョブ実行の最中に、直前の計算結果をフォック行列情報格納領域Bから読み出してホスト計算機に送信し、また、次のタイプBの計算ジョブで用いる情報をホスト計算機から受信して密度行列情報格納領域Bに書込むことが可能である。タイプBの計算ジョブの実行中にも、領域Aを用いて同様のデータ送受信動作を行なうことができる。

【0132】このようなダブル・バッファリング機構を用いることにより、データ送受信処理を計算ジョブの処理中に済ませることができれば、通信処理を計算処理で隠蔽することが可能となり、効率的な処理が行なえる。以下に示す計算例では、いずれもこのダブル・バッファリングを行なっている。

【0133】次に、密度行列情報の構成を図6および図7を用いて説明する。

【0134】図6に、ホスト計算機11からプロセッサエレメント14へ送信する密度行列情報のフォーマットの一例を示す。縮約シェルRの番号から縮約シェルW[Nw-1]の番号までは整数型データで、縮約シェルV[0]に関わる密度行列データブロックから縮約シェルW[Nw-1]に関わる密度行列データブロックは、1つまたは複数の浮動小数点型データ、または、固定小数点型データにより構成されるブロックである。

【0135】図6の上から2つの縮約シェルRおよび縮約シェルTの番号により、その密度行列情報を受け取ったプロセッサ・エレメント14が処理すべきRTペア番

10

20

30

40

50



号が決まる。

【0136】縮約シェルRおよびTの番号の次は、縮約シェルの集合Vおよび縮約シェルの集合Wの要素数を表わす2つの整数型データN<sub>v</sub>およびN<sub>w</sub>である。縮約シェルVは、それに含まれる縮約基底と縮約シェルRに含まれる縮約基底の組み合わせのうち、少なくとも1つがa bでのカットオフで生き残るもの、すなわち縮約シェルSのループでSがとりうる縮約シェル番号である。

【0137】同様に、縮約シェルWは、それに含まれる縮約基底と縮約シェルRに含まれる縮約基底の組み合わせのうち、少なくとも1つがc dでのカットオフで生き残るもの、すなわち縮約シェルUのループでUがとりうる縮約シェル番号である。

【0138】図7に、密度行列データブロックの構成例を示す。これは、縮約シェルV[0]に関わる密度行列データブロックを代表例として示したものである。

【0139】縮約シェルの軌道量子数が1より大きい場合には、その縮約シェルに関わる密度行列データブロックは、さらに縮約基底に関わるサブブロックにより構成されることになる。図7に示した例は、縮約シェルV  
[0]を構成する縮約基底が、M0(V[0]), M1(V[0]), ..., M<sub>m</sub>(V[0])というよう  
に、m+1個ある場合であり、それぞれの縮約基底に対応したサブブロックがm+1個ある。

【0140】1つのサブブロックは、さらに2つの領域に別れる。1つは、縮約シェルRを構成する縮約基底I0(R)からI<sub>i</sub>(R)と、M0(V[0])とをインデックスとする密度行列要素の領域である。もう1つは、縮約シェルTを構成する縮約基底K0(T)からK<sub>k</sub>(T)と、M0(V[0])とをインデックスとする密度行列要素の領域である。ここで、i+1, k+1は、それぞれ縮約シェルR, Tを構成する縮約基底の個数であり、これらは、縮約シェルR, Tの軌道量子数に依存する。これらの領域に含まれる密度行列要素の個数は、縮約シェルRおよびTの軌道量子数によって決まり、したがって、1組の密度行列情報の中では、どのサブブロックも同じ大きさとなる。

【0141】なお、説明は省略するが、フォック行列情報の構成も同様であり、そのデータ量は密度行列情報と同じである。

【0142】[プロセッサ・エレメントへのジョブの割り当て方法の実施の形態の説明] [各ジョブの形成方法および番号付けについて] プロセッサ・エレメントに割り当てられるジョブは、上述の説明から判るように、計算処理のみではなく、ホスト計算機との通信処理をも含んだものとして構成される。

【0143】この実施の形態においては、計算処理と通信処理との和からなるジョブの大きさ(サイズ)のばらつきが小さく、できるだけ、ジョブのサイズが均等になるように、各ジョブを形成する。また、各プロセッサ・

エレメントが有するメモリの容量には制限があるため、そのメモリに保持する密度行列やフォック行列の要素数になるべく少なくなるように、各ジョブを形成する。このため、この実施の形態では、次のようにしている。

【0144】上述したように、この実施の形態においては、ジョブは、R Tペア単位でプロセッサ・エレメントに割り当てられる。そして、R Tペアが決まると、縮約シェルSおよびUのループを回る回数が決まる。つまり、Rが大きい場合には、SおよびUのループを回る回数が多くなり、Rが小さい場合には、SおよびUのループを回る回数が少なくなる。そこで、Rが大きいジョブには、軌道量子数の高いシェルを割り当て、Rが小さいジョブには、軌道量子数の低いシェルを割り当てるようにする。

【0145】このようにすれば、R TペアのRが大きいジョブの場合には、SおよびUのループを回る回数が増えるが、このジョブに割り当てられる縮約シェルの軌道量子数が低いので、その内部の縮約基底に関するループを回る回数が増える。逆に、Rが小さいジョブの場合には、SおよびUのループを回る回数が少なくなるが、縮約シェルの軌道量子数が高いのでその内部の縮約基底に関するループを回る回数が増える。したがって、R Tペアにより定まるジョブに依存した計算量のばらつきは小さくなっている。

【0146】また、R TペアのRの番号が大きいジョブの場合には、図6のN<sub>v</sub>, N<sub>w</sub>が大きくなるが、このジョブに割り当てられる縮約シェルの軌道量子数が低いので、密度行列情報のデータ量が多くなりすぎないようにされているため、その結果としてジョブに依存した通信量のばらつきが小さくなっている。

【0147】したがって、ジョブのサイズのばらつきを小さくすることができる。

【0148】また、R TペアのRの番号が大きい場合には、N<sub>v</sub>, N<sub>w</sub>が大きくなるため、サブブロックのデータ量を小さくすることで、全体の容量を抑制する必要がある。サブブロックのデータ量は、前述したように、軌道量子数に依存し、軌道量子数の低い縮約シェルに関わるサブブロックはデータ量が少なくなる。

【0149】この実施の形態では、R TペアのRの番号の大きい縮約シェルの軌道量子数は低くされる。逆に、R TペアのRの番号が小さい場合には、N<sub>v</sub>, N<sub>w</sub>が小さいため、サブブロックのデータ量を少なくする必要がないので、番号の小さい縮約シェルの軌道量子数は高くされている。

【0150】したがって、このように、軌道量子数の高いシェルが、番号が小さい縮約シェルに、軌道量子数の低いシェルが、番号が大きい縮約シェルに割り当てられることで、容量の小さいメモリであっても、大規模な分子軌道計算への適用が可能となる。

【0151】この実施の形態においては、ジョブは、R

Tペア単位でプロセッサ・エレメントに割り当てられるので、ジョブ番号はRTペア番号と一致する。RTペア番号は、前述したように、任意に決定することができるが、例えば、図39の(数式38)により決定すると、R、Tの大きさの小さいものから順に、ジョブ番号が付与されることになる。

【0152】〔ジョブ割り当て方法の第1の実施の形態〕この例では、図2に示した計算機システムを用いた。そして、この例では、3残基ペプチドであるGlycine-Alanine-Glutamine(以下、GAQと記す。分子式： $C_{10}H_{18}N_4O_5$ )を計算の対象とし、基底関数として、6-311++G(3d, 2p)を用いた。

【0153】この例では、縮約シェルの数Nshellが336で、そのうち、軌道量子数が0のs軌道のシェルが167、軌道量子数が1のp軌道のシェルが112、軌道量子数が2のd軌道のシェルが57である。

【0154】この例では、縮約シェル番号を、0から56はd軌道のシェルに、57から168はp軌道のシェルに、169から335はs軌道のシェルに割り当てた。

【0155】さらに、この例では、RTペアの番号、すなわち、ジョブ番号は、図39の(数式38)により決め、RTペア番号の小さい順にプロセッサ・エレメントへのジョブ割り当てを行なった。

【0156】この例により、RT並列アルゴリズムで、SCFループを1回だけ実行した場合に要する処理時間とプロセッサ・エレメント数との関係の一例は、図8に示すようなものとなった。

【0157】図8において、破線で示した直線は、プロセッサ・エレメント数に反比例して処理時間が減少する理想的な場合を示している。この第1の例では、プロセッサ・エレメント数が増加して100の付近になると、理想直線から離れ、スケラブルな並列処理でなくなっていることが判る。

【0158】また、図9には、この第1の例において、全体の処理時間のうち、プロセッサ・エレメントが処理を行っていた時間の占める割合(稼働率)を示したもので、プロセッサ数が100の付近から稼働率が低下している。

【0159】この原因は、プロセッサ・エレメント数の増加に連れて、システム全体の計算処理能力が向上するのに対して、ホスト計算機とプロセッサ・エレメントとの間の通信性能が一定のままで通信量が増大することにより、通信時間が計算時間を越え、プロセッサ・エレメントが、データ受信待ちで計算を行なえない時間が増大するためであると考えられる。

【0160】スケラブルでなくなるもう1つの要因として考えられるのは、並列プロセッサへの計算負荷の割り当てが不均一となっていることである。しかしなが

ら、図10に示す全プロセッサ・エレメントにおける処理終了時間の分布を見ると、計算負荷の割り当ては、ほぼ均一に行なわれていることがわかる。

【0161】並列プロセッサ・エレメントにおける計算時間の合計は、36389秒であり、これを100プロセッサに均等に割り当てると、363.89秒の計算処理となる。これに対し、通信時間の合計は363.02秒であり、計算時間とほぼ等しくなっている。

【0162】しかしながら、図10に示すように、第1の例の並列処理における各プロセッサ・エレメントにおける計算時間は、420秒程度となっている。したがって、第1の例における並列処理は、効率の悪いものであると言える。

【0163】図11に、通信処理と単一のプロセッサ・エレメントにおける計算処理の進行状況を、処理効率の悪い場合と良い場合との比較で模式的に示す。図11において、網点部が処理を行なっている時間帯である。この網点部の合計の長さは通信と計算とで同じであり、また、効率の悪い場合と良い場合とでも同じである。

【0164】効率の悪い処理の場合には、図中、空白部分として示されるように、通信にも計算にも処理を行っていない時間帯が生じる。そのために、全体としての処理終了までの時間が長くなる。それに対し、効率の良い処理を行なえば、通信処理も計算処理も休みなく行なわれ、処理終了までの時間が短くなる。

【0165】上述の第1の例において、ジョブ1つあたりに発生する計算時間と通信時間との関係を図12に示す。この図12は、全56616ジョブから、ジョブ番号に関して等間隔に100個のジョブを抽出してプロットしたものである。図12に、参照インデックスとして示したように、ジョブにおける縮約シェルRおよびTの軌道量子数の組み合わせに応じて、プロットの形状を変えている。例えば、縮約シェルRの軌道量子数が1すなわちp軌道で、縮約シェルTの軌道量子数が2すなわちd軌道の場合の組み合わせ(pd)は、黒四角の点としてプロットした。

【0166】この図12から判るように、ジョブのサイズのばらつきは小さくても、ジョブによって通信時間と計算時間との割合(比)が異なっている。図12中の一点鎖線は、通信時間と計算時間の比が0.01となるラインである。プロセッサ・エレメント数が100の場合には、通信時間と計算時間との比が、この線より下のジョブが多いと通信量が多くなり、計算処理に待ちが生じることになる。逆に、通信時間と計算時間との比が、この線より上のジョブが多いと計算量が多くなり、通信処理を隠蔽しやすくなる。

【0167】前述のように、全体の通信時間と計算時間がほぼ等しい場合には、どの時刻においても、その時点で並列に行われているジョブに関わる通信量と計算量との比の平均値が一点鎖線上に乗るように、ジョブの割り

当てを制御すると、効率のよい並列処理が行える。

【0168】なお、この例では、縮約シェル番号の決め方を、前述のように、軌道量子数の高いシェルを小さい番号に、軌道量子数の低いシェルを大きい番号に割り当てるようにして、Rが大きい場合に密度行列情報のデータ量が多くなりすぎないようにしているため、その結果としてジョブ番号に依存した通信量のばらつきが小さくなっている。また、前述したように、上述のような縮約シェル番号の決め方をした場合には、ジョブ番号に依存した計算量のばらつきも小さくなっている。

【0169】縮約シェル番号を、軌道量子数の高いシェルを小さい番号に、軌道量子数の低いシェルを大きい番号に割り当てるようにしない場合には、計算時間や通信時間のジョブ番号に依存したばらつきが、図12に示したよりも大きくなり、複数のプロセッサ・エレメントに均等に負荷を割り当てることが、より困難になる。

【0170】図12に示したジョブ当たりの計算時間と通信時間の関係を調べると、一点鎖線よりも上の領域にあるのは、縮約シェルR、Tともにs軌道の場合の全てと、縮約シェルRがs軌道で、縮約シェルTがp軌道の場合の一部であることがわかる。

【0171】もう少し詳細に調べると、図39の(数式38)により決定したジョブ番号が、約35000より大きい場合には、プロットが一点鎖線よりも上にあり、それよりもジョブ番号が小さい場合には、下にあることがわかる。つまり、RTペア番号の小さいジョブは、通信時間の計算時間に対する比が大きくなり、RTペア番号の大きいジョブは、通信時間の計算時間に対する比が小さい。

【0172】以下に説明する第2の実施の形態から第5の実施の形態では、上述のようにして、計算時間や通信時間のばらつきが小さくなるように形成されると共に、図39の(数式38)により決定されたジョブ番号のジョブに、上述した図12に示す性質があることを利用して、より効率の良いジョブの割り当てを行なう。

【0173】[ジョブ割り当て方法の第2の実施の形態] 上記の考察から、番号の小さいジョブと、番号の大きいジョブとを混合させて割り当てることにより、通信負荷の時間的分散が行なえるものと期待される。そこで、この第2の実施の形態では、図13に示すような、割り当て待ち行列を作成して、並列プロセッサ・エレメント2あるいは14への割り当てを行う。

【0174】図13に示した例においては、説明を簡単にするために、全ジョブ数Nが33の場合の割り当て待ち行列を示したものである。図13において、括弧の外の数字がジョブ番号で、これが分子軌道計算におけるRT並列アルゴリズムではRTペア番号に相当する。また、括弧内の数字は、ジョブ割り当てを行なう順序を示している。

【0175】ホスト計算機は、この待ち行列を参照し

て、ジョブ番号を、この待ち行列の上から順に読み出しながら、プロセッサ・エレメントへのジョブの割り当てを行なう。

【0176】ここで、最初のジョブは、番号が0のジョブとし、それ以降は、前の番号をJとすると、

・JがN/2以下なら、番号がN-J-1のジョブ

・JがN/2以上なら、番号がN-Jのジョブ

を、プロセッサ・エレメントに次に割り当てるジョブとする。

10 【0177】このようにすることにより、RTペア番号の小さい、通信時間と計算時間との比の大きいジョブと、RTペア番号の大きい、通信時間と計算時間との比の小さいジョブとが交互に割り当てられるようになる。したがって、複数のプロセッサ・エレメントで並列に行なわれているジョブに関わる通信量と計算量との比の平均が、プロセッサ・エレメントの数分の1、プロセッサ・エレメントの数が100の場合には0.01、に近くなり、効率的な並列処理が行なえる。

20 【0178】図14に、第2の実施の形態のジョブの割り当て方法による並列処理方法の計算における処理時間とプロセッサ・エレメント数との関係を示す。この例の計算の対象とした分子、用いた基底関数は、ともに図8に結果を示した、第1の実施の形態におけるものと同じで、それぞれ、GAQと、6-311++G(3d, 2p)である。

【0179】図8の結果と比較して、プロセッサ・エレメント数が100付近における計算時間が、約9%短くなり、前述の第1の実施の形態の場合より高速な処理が可能となった。

30 【0180】図15は、この第2の実施の形態の場合におけるプロセッサ・エレメントの稼働率とプロセッサ・エレメント数との関係を示したもので、第1の実施の形態の説明における図9に対応するものである。

【0181】両者の比較から判るように、第2の実施の形態の場合には、プロセッサ・エレメント数が100の付近において、第1の実施の形態の例と比較して稼働率が高くなっている。

40 【0182】また、この第2の実施の形態において、プロセッサ・エレメント数が100の場合における、各プロセッサ・エレメントの処理終了時間を図16に示す。これは、第1の実施の形態の図10に対応するものである。並列プロセッサ・エレメントに均等に負荷が分配され、終了時間のばらつきが小さくなっていることが判る。

【0183】なお、割り当て待ち行列の先頭のジョブを、RTペア番号が最大のジョブとし、それ以降は、前の番号をJとした場合には、

・JがN/2以下なら、番号がN-Jのジョブ

・JがN/2以上なら、番号がN-J-1のジョブ

50 を、プロセッサ・エレメントに次に割り当てるジョブと

する。この場合にも、上述と同様の効果が得られる。

【0184】また、任意の番号のジョブを先頭とし、それ以降は、前の番号を  $J$  とすると、

・  $J$  が  $N/2$  以下なら、番号が  $N - J - 1$  のジョブ

・  $J$  が  $N/2$  以上なら、番号が  $N - J$  のジョブ

を次に割り当てるジョブとし、その番号が 0 未満または最大の  $RT$  ペア番号よりも大きくなったら最大の  $RT$  ペア番号の半分の番号のジョブから再スタートし、同様の手続きで次のジョブ番号を決める、などの類似の手法を用いて割り当て待ち行列を作成することによっても、同様の効果が得られる。

【0185】なお、第 2 の実施の形態において、次に割り当てるジョブの番号を決める際の条件に記述に用いている除算は、商を小数点以下まで求める正確な除算であり、第 3 の実施の形態以降で用いている整数の除算とは異なるものである。

【0186】〔ジョブ割り当て方法の第 3 の実施の形態〕第 3 の実施の形態では、複数個のプロセッサ・エレメントをグループ分けすると共に、それに対応して、複数個のジョブも同数のグループに分け、各グループに対して、それぞれ割り当て待ち行列を用意する。

【0187】すなわち、複数個のプロセッサ・エレメントは、グループ内のプロセッサ・エレメントの数がほぼ等しい  $G$  個 ( $G$  は 2 以上の整数) のプロセッサ・グループに分割すると共に、複数個のジョブを、グループ内のジョブの数がほぼ等しい、プロセッサ・グループの数と同数の  $G$  個のジョブ・グループに分割する。

【0188】そして、プロセッサ・グループと、ジョブ・グループとを 1 対 1 に対応させて、一つのジョブ・グループ内のジョブを、対応する一つのプロセッサ・グループ内のプロセッサ・エレメントに割り当てるようにする。

【0189】その割り当て方法としては、まず、通信時間と演算時間とからなるジョブサイズと、通信時間と演算時間との比とが近似するジョブ同士は、ジョブ・グループの異なるものにそれぞれ属するようにする割り当てる。さらに、ジョブサイズと、通信時間と演算時間との比とが近似するジョブは、他のプロセッサ・グループでの割り当て順序と重ならないように、複数のジョブ・グループのそれぞれ内での割り当て順番を、それぞれのジョブ・グループにおいて互いに異なるように割り当てる。

【0190】グループ数  $G$  が 4、ジョブ数  $N$  が 33 の簡単な場合について、図 17 および図 18 を用いて、この第 3 の実施の形態の場合の複数個のジョブ・グループに 1 対 1 に対応する複数個の待ち行列の作成方法の具体例を説明する。

【0191】前述例と同様に、これらの図 17、図 18 において、括弧外の番号はジョブ番号すなわち  $RT$  ペア番号を示し、括弧内の番号は同じ待ち行列に割り当てら

れたジョブ・グループ内での割り当てを行なう順序を示す。

【0192】まず、図 17 に示すように、ジョブ番号  $J$  を、ジョブ・グループ数 (待ち行列数)  $G$  で除算した時の余りに相当する番号  $g$  のグループに、各ジョブを分配する。ここで、各待ち行列内で、一旦、この分配されたジョブを番号順に並べる。

【0193】次に、ジョブ番号  $J$  をグループ数  $G$  で除算 (小数点以下を切り捨てる整数の除算を使用、以下同様) した時の商によりブロック分けを行う。図 17 の例では、各ブロック番号  $b$  は、その商である。

【0194】ここで、各ブロック番号  $b$  に属する複数のジョブは、ジョブ番号が近接したものであり、図 12 に示したように、互いにジョブサイズが近似し、かつ、計算時間と計算時間との比が近似したものとなり、それらのジョブが、それぞれ異なる待ち行列に分配されることになる。

【0195】全ジョブ数  $N$  をグループ数  $G$  で除算した時の商を  $B$  とした時、全ジョブ数  $N$  がグループ数  $G$  の倍数の場合には、ブロック数は  $B$  に、そうでない場合には、 $B + 1$  となる。後者の場合には、最後のブロックにジョブ番号のある待ち行列と、最後のブロックにジョブ番号がない待ち行列とが存在する。

【0196】次に、 $g$  番目のジョブ・グループ ( $g$  番目の待ち行列) 内のジョブのうち、 $B$  を、グループ数  $G$  で除算した商を  $x$  とし、 $x \times G$  以上、 $x \times (g + 1)$  未満の番号のジョブ・ブロックに含まれるジョブを、最初のジョブとする。これは、同じブロック内のジョブが、複数個のジョブ・グループで同じ順番とならないようにするためである。

【0197】図 14 の例では、 $B = N / G = 33 / 4 = 8$  であり、 $x = B / G = 8 / 4 = 2$  であるから、待ち行列 0 番のジョブ・グループであれば、 $x \times g = 2 \times 0 = 0$  以上、 $x \times (g + 1) = 2 \times 1 = 2$  未満の番号 0 または 1 のブロックから、開始ジョブを選択することになる。この例では、上記条件に適合する最小の番号を、開始ジョブとして選択することとした。

【0198】以下に示す計算例でもそのようにしたが、条件に適合する番号から任意に開始番号を選択しても良い。

【0199】次に、各ジョブ・グループにおいて、直前に割り当てが済んだジョブが  $b$  番目のジョブ・ブロックに含まれる場合には、 $b + 1$  番目のジョブ・ブロック内のジョブを次のジョブ番号とする。但し、 $b + 1$  番目のジョブ・ブロック内に  $g$  番目のグループ内のジョブがなければ、0 番目のジョブ・ブロック内のジョブを次のジョブとする。

【0200】図 14 の例における、待ち行列 1 番のジョブ・グループでは、 $b = 6$  番目のブロックに含まれる番号 25 のジョブの次のジョブは、 $b + 1 = 7$  番目のプロ

ックに含まれる番号 29 のジョブ、その次のジョブは（8 番目のブロックに含まれる 1 番目のグループのジョブがないので）0 番目のブロック内の番号 1 のジョブとなる。

【0201】なお、この例では、次のジョブを  $b+1$  番目のジョブ・ブロック内のジョブとしたが、次のジョブを  $b-1$  番目のジョブ・ブロック内のジョブとしてもよい。この場合、 $b-1$  が負となる時には、 $g$  番目のグループのジョブが含まれる最大の番号のジョブ・ブロック内のジョブを次のジョブとする。

【0202】以上のように順序を決め、各ジョブ・グループ内の割り当て順序に従ってジョブ番号を並べ直した、各ジョブ・グループの待ち行列を図 18 に示す。

【0203】図 19 に、第 3 の実施の形態の計算における処理時間とプロセッサ・エレメント数との関係を示す。

【0204】計算の対象とした分子、用いた基底関数は、第 1 の実施の形態で用いたものと同じである。なお、各プロセッサ・エレメントの演算処理能力は互いに等しいものとし、5 個のプロセッサ・エレメントで 1 つのプロセッサ・グループを構成した。

【0205】この図 19 から判るように、この第 3 の実施の形態によれば、図 8 の結果と比較して、プロセッサ・エレメント数が 100 付近における計算時間が約 12 % 短くなり、第 2 の実施の形態よりさらに高速な処理が可能となる。

【0206】図 20 は、この第 3 の実施の形態の場合の計算例におけるプロセッサ・エレメントの稼働率とプロセッサ・エレメント数との関係を示したもので、第 1 の実施の形態の説明における図 9 に対応するものである。これによれば、プロセッサ・エレメント数が 100 の付近において、第 1 の実施の形態と比較して稼働率が高くなり、100 % に近づいている。

【0207】また、この第 3 の実施の形態において、プロセッサ・エレメント数が 100 の場合における、各プロセッサ・エレメントの処理終了時間を図 21 に示す。終了時間のばらつきが 4 秒程度あるものの、並列プロセッサ・エレメントにほぼ均等に負荷が分配されていることが判る。

【0208】〔ジョブ割り当て方法の第 4 の実施の形態〕第 4 の実施の形態は、プロセッサ・エレメントをグループ分けし、各グループに対して割り当て待ち行列を用意する点では、前述の第 3 の実施の形態と同様であるが、プロセッサ・エレメントのグループ分けの方法が、第 3 の実施の形態と異なる。

【0209】グループ数  $G$  が 4、ジョブ数  $N$  が 33 の簡単な場合について、図 22 および図 23 を用いて、この第 4 の実施の形態の場合の複数のジョブ・グループのそれぞれに対応する待ち行列の作成方法を説明する。なお、図 22、図 23 において、括弧の外の番号はジョブ

番号すなわち  $RT$  ペア番号を示し、括弧内の番号は同じ待ち行列に割り当てられたジョブ・グループ内で割り当てを行なう順序を示すのは、前述の実施の形態の場合と同様である。

【0210】まず、図 22 に示すように、ジョブ番号  $J$  をジョブ・グループ数（待ち行列数） $G$  の 2 倍の  $2 \times G$  で除算した時の余りが  $g$  または  $2 \times G - g - 1$  に等しいジョブが、 $g$  番目のグループに含まれるように、各ジョブを分配する。そして、分配したジョブを、各待ち行列内で、一旦、番号順に並べる。

【0211】次に、ジョブ番号  $J$  をグループ数  $G$  で除算（小数点以下を切り捨てる整数の除算を使用、以下同様）した時の商によりブロック分けを行う。図 22 において、各ブロック番号  $b$  は、その商である。

【0212】全ジョブ数  $N$  をグループ数  $G$  で除算した時の商を  $B$  とした時、全ジョブ数  $N$  がグループ数  $G$  の倍数の場合には、ブロック数は  $B$  に、そうでない場合には  $B+1$  となる。後者の場合、最後のブロックにジョブ番号のある待ち行列と、最後のブロックにジョブ番号がない待ち行列とが存在する。

【0213】次に、 $g$  番目のジョブ・グループ（ $g$  番目の待ち行列）内のジョブのうち、 $B$  をグループ数  $G$  で除算した商を  $x$  として、 $x \times g$  以上、 $x \times (g+1)$  未満の番号のジョブ・ブロックに含まれるジョブを、最初のジョブとする。これは、同じブロック内のジョブが、複数のジョブ・グループで同じ順番とならないようにするためである。

【0214】図 22 の例では、 $B = N / G = 33 / 4 = 8$  であり、 $x = B / G = 8 / 4 = 2$  であるから、待ち行列 0 番目のグループであれば、 $x \times g = 2 \times 0 = 0$  以上、 $x \times (g+1) = 2 \times 1 = 2$  未満の番号 0 または 1 のブロックから、開始ジョブを選択することになる。この例では、上記条件に適合する最小の番号を、開始ジョブとして選択することとした。

【0215】以下に示す計算例でもそのようにしたが、条件に適合する番号から任意に開始番号を選択しても良い。

【0216】次に、各ジョブ・グループにおいて、直前に割り当てが済んだジョブが  $b$  番目のジョブ・ブロックに含まれる場合には、 $b+1$  番目のジョブ・ブロック内のジョブを次のジョブ番号とする。但し、 $b+1$  番目のジョブ・ブロック内に  $g$  番目のグループ内のジョブがなければ、0 番目のジョブ・ブロック内のジョブを次のジョブとする。

【0217】図 22 の例における待ち行列 1 番目のグループでは、 $b=6$  番目のブロックに含まれる番号 25 のジョブの次のジョブは、 $b+1=7$  番目のブロックに含まれる番号 30 のジョブ、その次のジョブは（8 番目のブロックに含まれる 1 番目のグループのジョブがないので）0 番目のブロック内の番号 1 のジョブとなる。

【0218】なお、この例では、次のジョブを、 $b+1$  番目のジョブ・ブロック内のジョブとしたが、次のジョブを、 $b-1$  番目のジョブ・ブロック内のジョブとしてもよい。この場合、 $b-1$  が負となる時には、 $g$  番目のグループのジョブが含まれる最大の番号のジョブ・ブロック内のジョブを次のジョブとする。

【0219】以上のように順序を決め、各ジョブ・グループ内の割り当て順序に従ってジョブ番号を並べ直した、各ジョブ・グループの待ち行列を図 23 に示す。

【0220】図 24 に、第 4 の実施の形態の計算における処理時間とプロセッサ・エレメント数との関係を示す。計算の対象とした分子、用いた基底関数は、第 1 の実施の形態で用いたものと同じである。なお、各プロセッサ・エレメントの演算処理能力は互いに等しいものとし、5 個のプロセッサ・エレメントで 1 つのプロセッサ・グループを構成した。

【0221】図 24 から判るように、この第 4 の実施の形態によれば、図 8 の結果と比較して、プロセッサ・エレメント数が 100 付近における計算時間が約 12% 短くなり、第 2 の実施の形態よりさらに高速な処理が可能となる。

【0222】図 25 は、この第 4 の実施の形態の場合の計算例におけるプロセッサ・エレメントの稼働率とプロセッサ・エレメント数との関係を示したもので、第 1 の実施の形態の説明における図 9 に対応するものである。これによれば、プロセッサ・エレメント数が 100 の付近において、第 1 の実施の形態と比較して稼働率が高くなり 100% に近づいている。

【0223】また、この第 4 の実施の形態において、プロセッサ・エレメント数が 100 の場合における、各プロセッサ・エレメントの処理終了時間を図 26 に示す。終了時間のばらつきが 3.5 秒程度あるものの、並列プロセッサ・エレメントにほぼ均等に負荷が分配されている。

【0224】〔ジョブ割り当て方法の第 5 の実施の形態〕第 5 の実施の形態は、プロセッサ・エレメントおよびジョブを、1 対 1 にグループ分けし、各グループに対して割り当て待ち行列を用意する点では、第 3 および第 4 の実施の形態と同様であり、また、プロセッサ・エレメントのグループ分けの方法が第 3 の実施の形態と同様であるが、グループ内の割り当て順序の決め方が第 3 および第 4 の実施の形態と異なる。

【0225】グループ数  $G$  が 4、ジョブ数  $N$  が 65、グループ当たりのプロセッサ・エレメント数が 2 の簡単な場合について、図 27 および図 28 を用いて、この第 5 の実施の形態の場合の複数のジョブ・グループのそれぞれに対応する待ち行列の作成方法を説明する。なお、これらの図 27、図 28 において、括弧の外の番号はジョブ番号すなわち RT べア番号を示し、括弧内の番号は同じ待ち行列に割り当てられたジョブ・グループ内で割

り当てを行なう順序を示すのは、前述の実施の形態の場合と同様である。

【0226】まず、図 27 に示すように、ジョブ番号  $J$  をジョブ・グループ数（待ち行列数） $G$  で除算した時の余りが  $g$  に等しいジョブが、 $g$  番目のグループに含まれるように、各ジョブを分配する。この分配方法は、第 3 の実施の形態と同様であるが、第 4 の実施の形態と同様の方法で分配してもよい。ここで、分配したジョブを、各待ち行列内では、一旦、番号順に並べる。

【0227】次に、ジョブ番号  $J$  をグループ数  $G$  で除算（小数点以下を切り捨てる整数の除算を使用、以下同様）した時の商によりブロック分けを行う。図 22 において、各ブロック番号  $b$  は、その商である。

【0228】全ジョブ数  $N$  をグループ数  $G$  で除算した時の商を  $B$  とした時、全ジョブ数  $N$  がグループ数  $G$  の倍数の場合には、ブロック数は  $B$  に、そうでない場合には  $B+1$  となる。後者の場合、最後のブロックにジョブ番号のある待ち行列と、最後のブロックにジョブ番号がない待ち行列とが存在する。

【0229】次に、 $g$  番目のジョブ・グループ（ $g$  番目の待ち行列）内のジョブのうち、 $B$  を  $G$  で除算した商を  $x$  とし、さらに  $x$  をグループに割り当てられるプロセッサ・エレメントの数  $m$  で除算した商を  $y$  として、 $y \times g$  以上、 $y \times (g+1)$  未満の番号のジョブ・ブロックに含まれるジョブを、最初のジョブとする。これは、同じブロック内のジョブが、複数のジョブ・グループで同じ順番とならないようにするためである。

【0230】図 27 の例では、 $B=N/G=65/4=16$  であり、 $x=B/G=16/4=4$ 、 $y=x/m=4/2=2$  であるから、待ち行列 0 番目のグループであれば、 $y \times g=2 \times 0=0$  以上、 $y \times (g+1)=2 \times 1=2$  未満の番号 0 または 1 のブロックから、開始ジョブを選択することになる。この例では、上記条件に適合する番号のうち最も小さい番号を、開始ジョブとして選択した。

【0231】以下に示す計算例でも上記条件に適合する最小の番号を選択するようにしたが、条件に適合する番号から任意に開始番号を選択しても良い。また、開始ジョブを選択するブロックの条件を、第 3 の実施の形態あるいは第 4 の実施の形態と同様にしてもよい。

【0232】次に、各ジョブ・グループにおいて、直前に割り当てが済んだジョブが  $b$  番目のジョブ・ブロックに含まれる場合には、 $b+B/m$  番目のジョブ・ブロック内のジョブを次のジョブ番号とする。但し、 $b+B/m$  番目のジョブ・ブロック内に  $g$  番目のグループ内のジョブがなければ、 $b+1$  を  $(B/m)$  で除算した余りに等しい番号のジョブ・ブロック内のジョブを次のジョブとする。

【0233】図 27 の例における待ち行列 1 番目のグループでは、 $b=7$  番目のブロックに含まれる番号 29 の

ジョブの次のジョブは、 $b + (B/m) = 7 + (16/2) = 15$  番目のブロックに含まれる番号 61 のジョブとなる。その次のジョブは、 $b = 15$  として、 $b + (B/m) = 15 + (16/2) = 23$  番目のブロックに含まれる 1 番目のグループのジョブがないので、 $b + 1 = 16$  を、 $(B/m) = 8$  で除算した余りに等しい 0 番目のブロック内の番号 1 のジョブとなる。

【0234】なお、この例では、次のジョブを、 $b + B/m$  番目のジョブ・ブロック内のジョブとしたが、次のジョブを、 $b - B/m$  番目のジョブ・ブロック内のジョブとしてもよい。この場合、 $b - B/m$  が負で、かつ、 $b$  が 0 であれば、 $(B/m)$  で除算した余りが  $(B/m) - 1$  に等しくなる番号のジョブ・ブロックのうち、 $g$  番目のグループのジョブを含む最大の番号のジョブ・ブロック内のジョブを次のジョブとし、 $b - B/m$  が負で、かつ、 $b$  が 0 でなければ、 $(B/m)$  で除算した余りが  $b - 1$  を  $(B/m)$  で除算した余りと等しくなるようなジョブ・ブロック番号のうち  $g$  番目のジョブ・グループのジョブを含む最大の番号のジョブ・ブロック内のジョブを次のジョブとする。

【0235】このように順序を決め、グループ内の割り当て順序に従ってジョブ番号を並べ直したのが、図 28 に示す待ち行列である。

【0236】図 29 に、第 5 の実施の形態の計算における処理時間とプロセッサ・エレメント数との関係を示す。計算の対象とした分子、用いた基底関数は、第 1 の実施の形態で用いたものと同じである。なお、各プロセッサ・エレメントの演算処理能力は互いに等しいものとし、5 個のプロセッサ・エレメントで 1 つのプロセッサ・グループを構成した。

【0237】図 29 から判るように、この第 5 の実施の形態によれば、図 8 の結果と比較して、プロセッサ・エレメント数が 100 付近における計算時間が約 13% 短くなり、第 3 の実施の形態および第 4 の実施の形態よりさらに高速な処理が可能となった。

【0238】図 30 は、この計算例におけるプロセッサ・エレメントの稼働率とプロセッサ・エレメント数との関係を示したもので、第 1 の実施の形態の説明における図 9 に対応するものである。これによれば、プロセッサ・エレメント数が 100 の付近において、従来例と比較して稼働率が高くなりほぼ 100% となっている。

【0239】また、この第 5 の実施の形態において、プロセッサ・エレメント数が 100 の場合における、各プロセッサ・エレメントの処理終了時間を図 31 に示す。終了時間のばらつきが 1.5 秒程度と小さく、並列プロセッサ・エレメントに均等に負荷が分配されていることが確認できた。

【0240】以上に説明した第 3、第 4、第 5 の実施の形態においては、同一の演算処理性能を有する 5 個のプロセッサ・エレメントによりプロセッサ・グループを構

成したが、プロセッサ・エレメントの個数はこれに限る必要はない。

【0241】また、全てのプロセッサ・エレメントの演算処理性能が同じである必要もなく、例えば 1 つのプロセッサ・グループを 2 倍の演算処理性能を有するプロセッサ・エレメント 2 つで構成し、他のプロセッサ・グループを 1 倍の演算処理性能を有するプロセッサ・エレメント 4 つで構成するなど、プロセッサ・グループ内の合計の演算処理性能がグループ間で同一となっていれば良い。

【0242】また、上述の実施の形態では、非経験的分子軌道計算において RT 並列アルゴリズムを用いる場合を例として、この発明の説明を行なったが、他の並列処理においても、この発明を適用することが可能である。

【0243】

【発明の効果】以上説明したように、この発明による並列処理方法における各プロセッサへのジョブ割り当て方法を用いると、効率の良い並列処理が可能となり、システム全体の処理時間を短くすることができる。

【図面の簡単な説明】

【図 1】この発明の実施の形態で用いる計算機システムの一例を示すブロック図である。

【図 2】この発明の実施の形態で用いる計算機システムの他の例を示すブロック図である。

【図 3】この発明の実施の形態の RT 並列アルゴリズムをプログラム・コード形式で示す図である。

【図 4】この発明の実施の形態の RT 並列アルゴリズムのフローチャートを示す図である。

【図 5】この発明の実施の形態で用いるデータのダブル・バッファリングを行なう場合のメモリ領域の使い方を説明するための図である。

【図 6】この発明の実施の形態において、ホスト計算機からプロセッサエレメントへ送信される密度行列情報のフォーマットの一例を示す図である。

【図 7】この発明の実施の形態における密度行列データブロックの構成例を示す図である。

【図 8】この発明の第 1 の実施の形態の分子軌道計算における並列プロセッサ数と SCF 計算の処理時間の関係を示す図である。

【図 9】この発明の第 1 の実施の形態の分子軌道計算における並列プロセッサ数とプロセッサの稼働率の関係を示す図である。

【図 10】この発明の第 1 の実施の形態の分子軌道計算における個々の並列プロセッサの処理の終了時間を示す図である。

【図 11】効率の悪い処理方法と効率の良い処理方法における通信処理と計算処理の進行状況を示す模式図である。

【図 12】この発明の第 1 の実施の形態の分子軌道計算における、ジョブあたりに発生する通信量と計算量の相

関関係を示す図である。

【図 13】この発明の第 2 の実施の形態で用いる、プロセッサ・エレメントへのジョブの割り当てのための待ち行列の例を示す図である。

【図 14】この発明の第 2 の実施の形態における、並列プロセッサ数と S C F 計算の処理時間の関係を示す図である。

【図 15】この発明の第 2 の実施の形態における、並列プロセッサ数とプロセッサの稼働率の関係を示す図である。

【図 16】この発明の第 2 の実施の形態における、個々の並列プロセッサの処理の終了時間を示す図である。

【図 17】この発明の第 3 の実施の形態におけるジョブ・グループ分割方法およびジョブの割り当て方法を説明するための図である。

【図 18】この発明の第 3 の実施の形態における、プロセッサ・エレメントへのジョブの割り当てのための待ち行列の例を示す図である。

【図 19】この発明の第 3 の実施の形態における、並列プロセッサ数と S C F 計算の処理時間の関係を示す図である。

【図 20】この発明の第 3 の実施の形態における、並列プロセッサ数とプロセッサの稼働率の関係を示す図である。

【図 21】この発明の第 3 の実施の形態における、個々の並列プロセッサの処理の終了時間を示す図である。

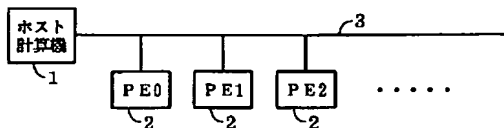
【図 22】この発明の第 4 の実施の形態におけるジョブ・グループ分割方法およびジョブの割り当て方法を説明するための図である。

【図 23】この発明の第 4 の実施の形態における、プロセッサ・エレメントへのジョブの割り当てのための待ち行列の例を示す図である。

【図 24】この発明の第 4 の実施の形態における、並列プロセッサ数と S C F 計算の処理時間の関係を示す図である。

【図 25】この発明の第 4 の実施の形態における、並列プロセッサ数とプロセッサの稼働率の関係を示す図である。

【図 1】



る。

【図 26】この発明の第 4 の実施の形態における、個々の並列プロセッサの処理の終了時間を示す図である。

【図 27】この発明の第 5 の実施の形態におけるジョブ・グループ分割方法およびジョブの割り当て方法を説明するための図である。

【図 28】この発明の第 5 の実施の形態における、プロセッサ・エレメントへのジョブの割り当てのための待ち行列の例を示す図である。

【図 29】この発明の第 5 の実施の形態における、並列プロセッサ数と S C F 計算の処理時間の関係を示す図である。

【図 30】この発明の第 5 の実施の形態における、並列プロセッサ数とプロセッサの稼働率の関係を示す図である。

【図 31】この発明の第 5 の実施の形態における、個々の並列プロセッサの処理の終了時間を示す図である。

【図 32】原始基底関数と、その角運動量、軌道指数、原子核座標との対応例を示す図である。

【図 33】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 34】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 35】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 36】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 37】非経験的分子軌道計算法の説明に用いる数式を示す図である。

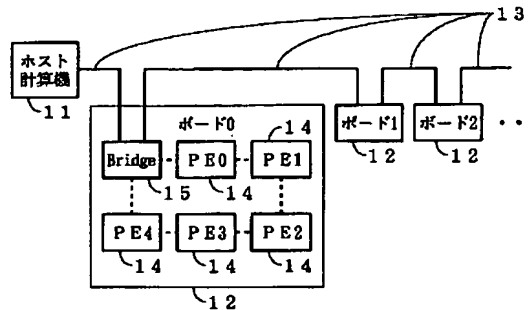
【図 38】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図 39】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【符号の説明】

- 1、11 ホスト計算機
- 2、14 プロセッサ・エレメント
- 3、13 バス

【図 2】



【図 13】

待ち行列
0(0)
32(1)
1(2)
31(3)
2(4)
⋮
15(30)
17(31)
16(32)



【図 3】

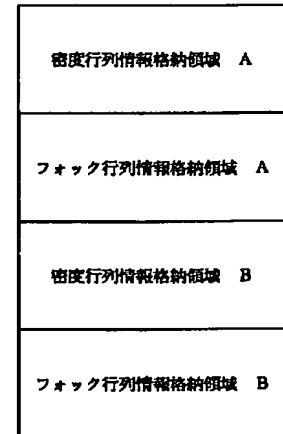
RT並列アルゴリズム (この発明の実施形態)

```

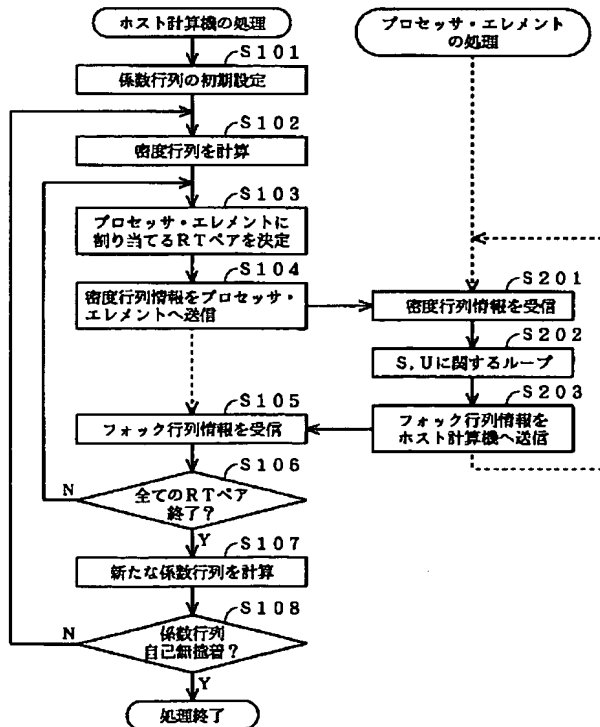
for(R=0; R<Nshell; R++){
for(T=0; T<=R; T++){
    for(S=0; S<=R; S++){
        for(U=0; U<=R; U++){
            for(I=b_basis(R); I<=e_basis(R); I++){
                for(J=b_basis(S); J<=e_basis(S); J++){
                    for(K=b_basis(T); K<=e_basis(T); K++){
                        for(L=b_basis(U); L<=e_basis(U); L++){
                            G_IJKL=G(I, J, K, L);
                            F[I][J]+=-P[K][L]*G_IJKL;
                            if(L<=K&&K<I){
                                if(J<I) F[K][L]+=2*P[I][J]*G_IJKL;
                                else F[K][L]+=P[I][J]*G_IJKL;
                            }
                            F[I][L]-=0.5*P[K][J]*G_IJKL;
                            if(J<I&&K<=J)
                                F[K][J]-=0.5*P[I][L]*G_IJKL;
                            if(K<I&&J<=K&&L<I)
                                F[K][J]-=0.5*P[I][L]*G_IJKL;
                        }
                    }
                }
            }
        }
    }
}
}

```

【図 5】

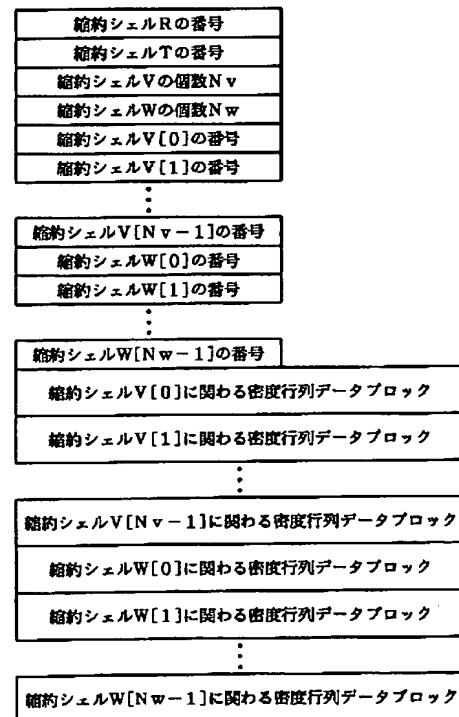


【図 4】



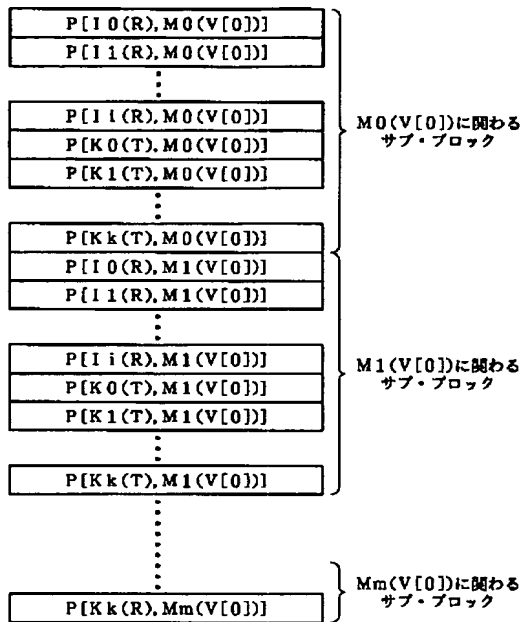
【図 6】

転送する密度行列情報のフォーマット例

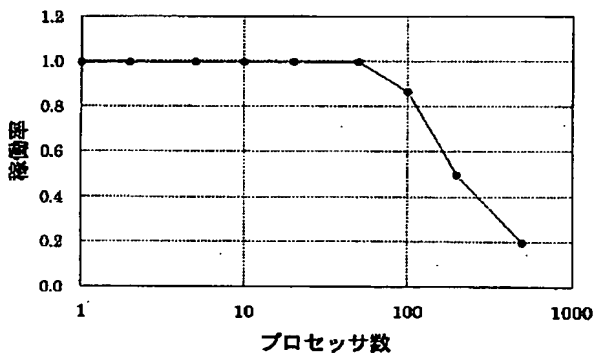


【図7】

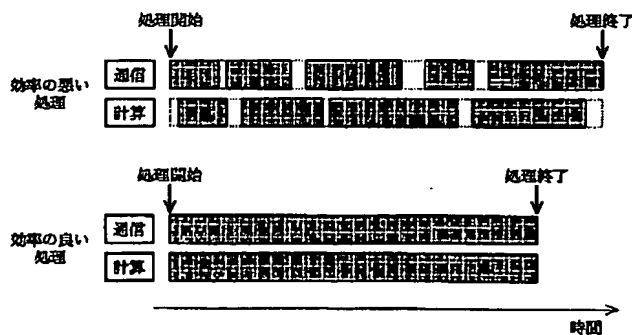
密着行列データブロックの構成例



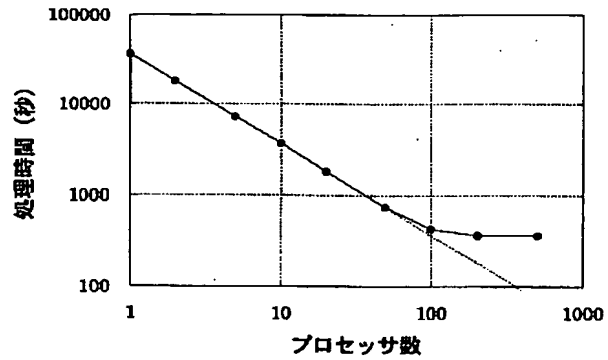
【図9】



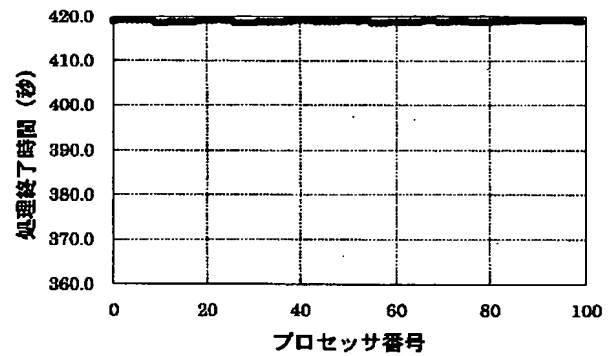
【図11】



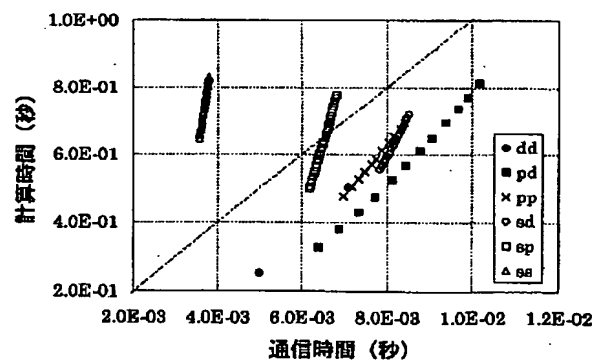
【図8】



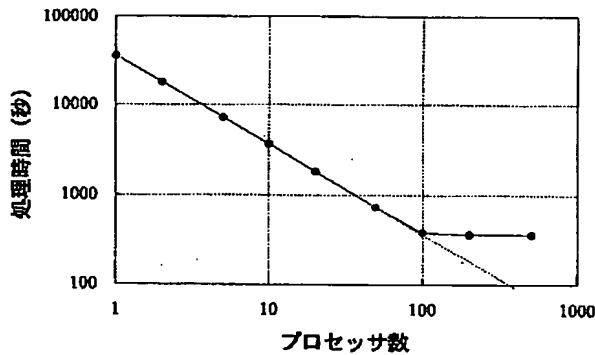
【図10】



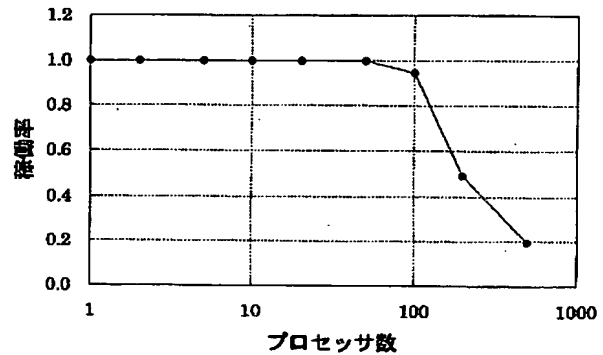
【図12】



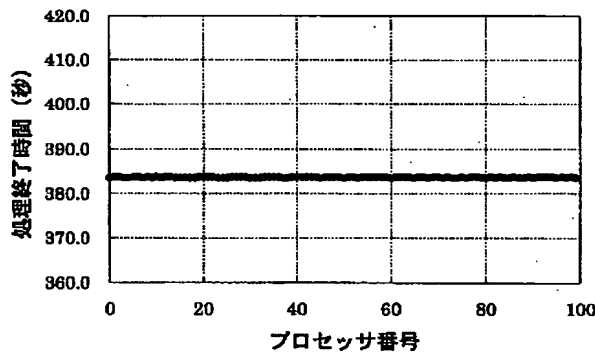
【図 14】



【図 15】



【図 16】



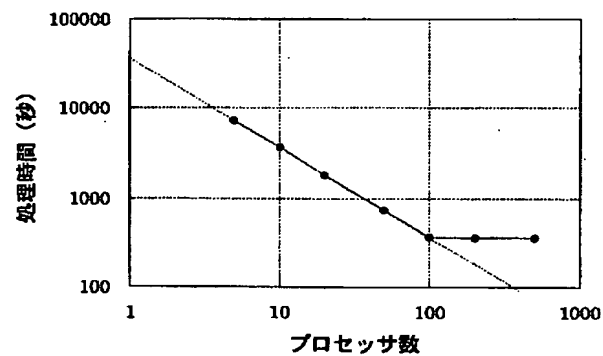
【図 17】

	待ち行列 0番のジョブ	待ち行列 1番のジョブ	待ち行列 2番のジョブ	待ち行列 3番のジョブ
ブロック0	0(0)	1(6)	2(4)	3(2)
ブロック1	4(1)	5(7)	6(5)	7(3)
ブロック2	8(2)	9(0)	10(6)	11(4)
ブロック3	12(3)	13(1)	14(7)	15(5)
ブロック4	16(4)	17(2)	18(0)	19(6)
ブロック5	20(5)	21(3)	22(1)	23(7)
ブロック6	24(6)	25(4)	26(2)	27(0)
ブロック7	28(7)	29(5)	30(3)	31(1)
ブロック8	32(8)			

【図 18】

待ち行列 0番	待ち行列 1番	待ち行列 2番	待ち行列 3番
0(0)	9(0)	18(0)	27(0)
4(1)	13(1)	22(1)	31(1)
8(2)	17(2)	26(2)	3(2)
12(3)	21(3)	30(3)	7(3)
16(4)	25(4)	2(4)	11(4)
20(5)	29(5)	6(5)	15(5)
24(6)	1(6)	10(6)	19(6)
28(7)	5(7)	14(7)	23(7)
32(8)			

【図 19】

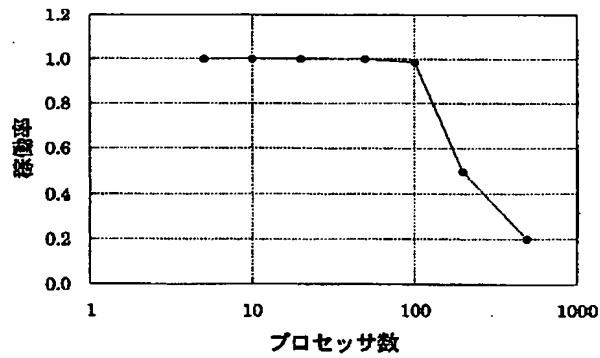


【図 32】

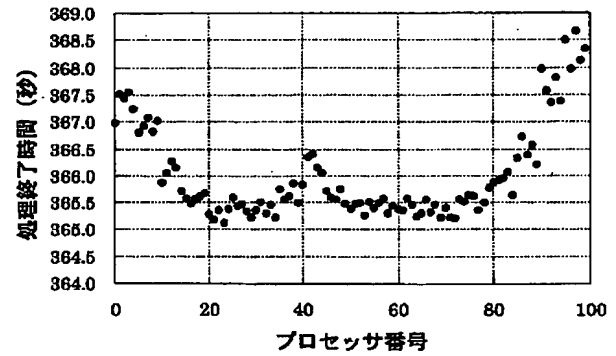
【表 1 原始基底関数  $i, j, k, l$  の角運動量、軌道指数、原子核座標】

原始基底関数の番号	角運動量	軌道指数	原子核座標
$i$	$a$	$\zeta_a$	A
$j$	$b$	$\zeta_b$	B
$k$	$c$	$\zeta_c$	C
$l$	$d$	$\zeta_d$	D

【図 20】



【図 21】



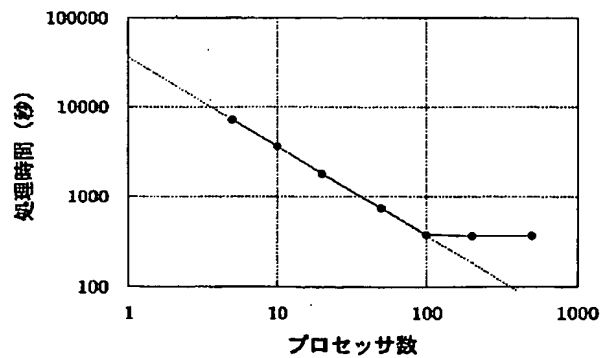
【図 22】

	待ち行列 0番のジョブ	待ち行列 1番のジョブ	待ち行列 2番のジョブ	待ち行列 3番のジョブ
ブロック0	0(0)	1(6)	2(4)	3(2)
ブロック1	7(1)	6(7)	5(5)	4(3)
ブロック2	8(2)	9(0)	10(6)	11(4)
ブロック3	15(3)	14(1)	13(7)	12(5)
ブロック4	16(4)	17(2)	18(0)	19(6)
ブロック5	23(5)	22(3)	21(1)	20(7)
ブロック6	24(6)	25(4)	26(2)	27(0)
ブロック7	31(7)	30(5)	29(3)	28(1)
ブロック8	32(8)			

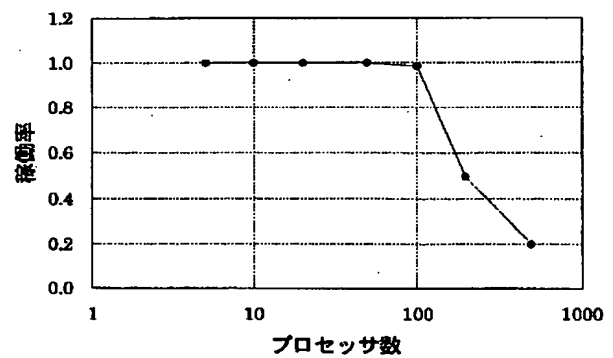
【図 23】

待ち行列 0番	待ち行列 1番	待ち行列 2番	待ち行列 3番
0(0)	9(0)	18(0)	27(0)
7(1)	14(1)	21(1)	28(1)
8(2)	17(2)	26(2)	3(2)
15(3)	22(3)	29(3)	4(3)
16(4)	25(4)	2(4)	11(4)
23(5)	30(5)	5(5)	12(5)
24(6)	1(6)	10(6)	19(6)
31(7)	6(7)	13(7)	20(7)
32(8)			

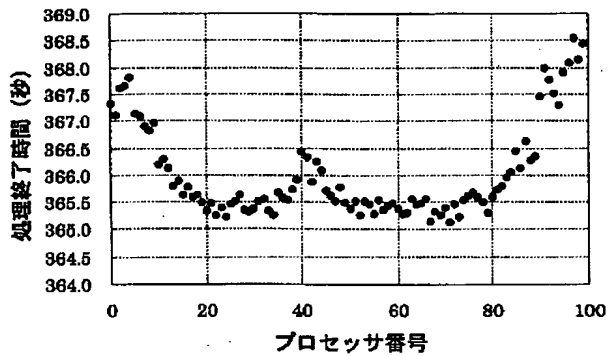
【図 24】



【図 25】



【図 26】



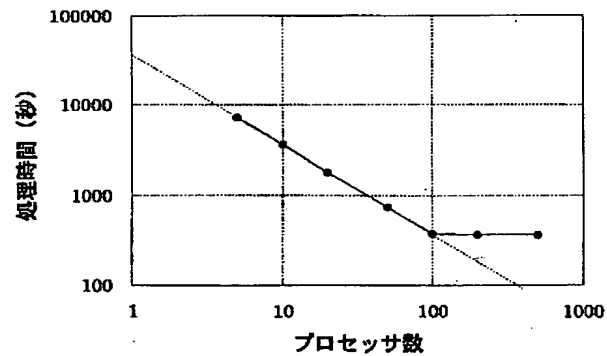
【図 27】

	待ち行列 0番のジョブ	待ち行列 1番のジョブ	待ち行列 2番のジョブ	待ち行列 3番のジョブ
ブロック0	0(0)	1(12)	2(8)	3(4)
ブロック1	4(3)	5(14)	6(10)	7(6)
ブロック2	8(5)	9(0)	10(12)	11(8)
ブロック3	12(7)	13(2)	14(14)	15(10)
ブロック4	16(9)	17(4)	18(0)	19(12)
ブロック5	20(11)	21(6)	22(2)	23(14)
ブロック6	24(13)	25(8)	26(4)	27(0)
ブロック7	28(15)	29(10)	30(8)	31(2)
ブロック8	32(1)	33(13)	34(9)	35(5)
ブロック9	36(4)	37(15)	38(11)	39(7)
ブロック10	40(6)	41(1)	42(13)	43(9)
ブロック11	44(8)	45(3)	46(15)	47(11)
ブロック12	48(10)	49(5)	50(1)	51(13)
ブロック13	52(12)	53(7)	54(3)	55(15)
ブロック14	56(14)	57(9)	58(5)	59(1)
ブロック15	60(16)	61(11)	62(7)	63(3)
ブロック16	64(2)			

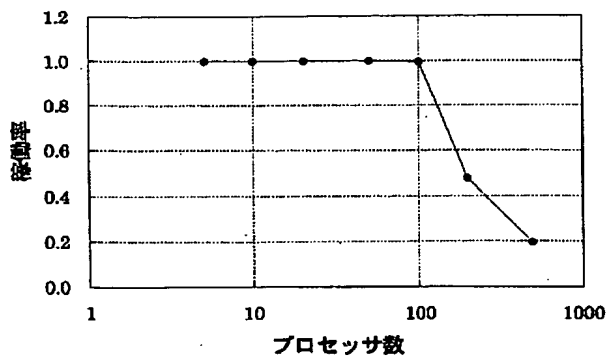
【図 28】

待ち行列 0番	待ち行列 1番	待ち行列 2番	待ち行列 3番
0(0)	9(0)	18(0)	27(0)
32(1)	41(1)	50(1)	59(1)
64(2)	13(2)	22(2)	31(2)
4(3)	45(3)	54(3)	63(3)
36(4)	17(4)	26(4)	3(4)
8(5)	49(5)	58(5)	35(5)
40(6)	21(6)	30(6)	7(6)
12(7)	53(7)	62(7)	39(7)
44(8)	25(8)	2(8)	11(8)
16(9)	57(9)	34(9)	43(9)
48(10)	29(10)	6(10)	15(10)
20(11)	61(11)	38(11)	47(11)
52(12)	1(12)	10(12)	19(12)
24(13)	33(13)	42(13)	51(13)
56(14)	5(14)	14(14)	23(14)
28(15)	37(15)	46(15)	55(15)
60(16)			

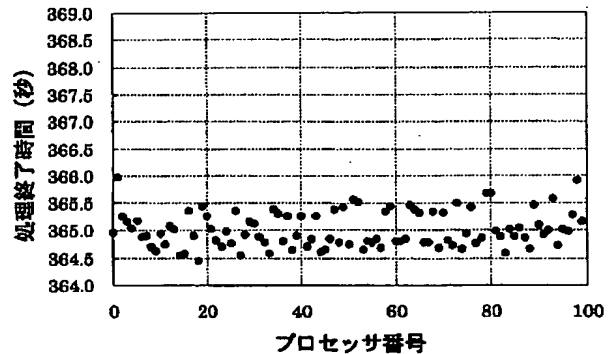
【図 29】



【図 30】



【図 31】



【図33】

(数式1)

$$\phi_{\mu} = \sum_i x_i c_{i\mu}$$

(数式2)

$$\Psi(1, 2, \dots, 2n) = \frac{1}{\sqrt{(2n)!}} \times$$

$$\begin{vmatrix} \phi_{1\alpha}(1) & \phi_{1\beta}(1) & \phi_{2\alpha}(1) & \phi_{2\beta}(1) & \cdots & \phi_{n\alpha}(1) & \phi_{n\beta}(1) \\ \phi_{1\alpha}(2) & \phi_{1\beta}(2) & \phi_{2\alpha}(2) & \phi_{2\beta}(2) & \cdots & \phi_{n\alpha}(2) & \phi_{n\beta}(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{1\alpha}(2n) & \phi_{1\beta}(2n) & \phi_{2\alpha}(2n) & \phi_{2\beta}(2n) & \cdots & \phi_{n\alpha}(2n) & \phi_{n\beta}(2n) \end{vmatrix}$$

(数式3)

$$H = H_1 + H_2$$

(数式4)

$$H_1 = \sum_p \left[ -\frac{1}{2} \nabla_p^2 - \sum_A \frac{Z_A}{r_{pA}} \right]$$

(数式5)

$$H_2 = \sum_p \sum_{q(>p)} \frac{1}{r_{pq}}$$

【図34】

(数式6)

$$\begin{aligned} \varepsilon &= \int \Psi H \Psi d\tau = \int \Psi (H_1 + H_2) \Psi d\tau \\ &= 2 \sum_{\mu} H_{\mu} + \sum_{\mu} \sum_{\nu} (2 J_{\mu\nu} - K_{\mu\nu}) \end{aligned}$$

(数式7)

$$H_{\mu} = \int \phi_{\mu}(1) h^{\text{core}}(1) \phi_{\mu}(1) d\tau_1$$

(数式8)

$$J_{\mu\nu} = \iint \phi_{\mu}(1) \phi_{\mu}(1) \frac{1}{r_{12}} \phi_{\nu}(2) \phi_{\nu}(2) d\tau_1 d\tau_2$$

(数式9)

$$K_{\mu\nu} = \iint \phi_{\mu}(1) \phi_{\nu}(1) \frac{1}{r_{12}} \phi_{\mu}(2) \phi_{\nu}(2) d\tau_1 d\tau_2$$

【図35】

(数式10)

$$\epsilon = 2 \sum_{\mu} \left[ \sum_I \sum_J c_{I\mu} c_{J\mu} H_{IJ} \right] \\ + \sum_{\mu} \sum_{\nu} \left\{ \sum_I \sum_J \sum_K \sum_L c_{I\mu} c_{J\mu} c_{K\nu} c_{L\nu} [2G(I, J, K, L) - G(I, K, J, L)] \right\}$$

(数式11)

$$H_{IJ} = \int x_I(1) h^{\text{core}}(1) x_J(1) d\tau_1$$

(数式12)

$$h^{\text{core}}(1) = -\frac{1}{2} \nabla_1^2 - \sum_A \frac{Z_A}{r_{1A}}$$

(数式13)

$$G(I, J, K, L) = \iint x_I(1) x_J(1) \frac{1}{r_{12}} x_K(2) x_L(2) d\tau_1 d\tau_2$$

【図36】

(数式14)

$$\sum_I (P_{IJ} - e_{\mu} S_{IJ}) C_{I\mu} = 0$$

(数式15)

$$F_{IJ} = H_{IJ} + \sum_K \sum_L P_{KL} \left[ G(I, J, K, L) - \frac{1}{2} G(I, K, J, L) \right]$$

(数式16)

$$\epsilon_{\mu} = H_{\mu} + \sum_{\nu} (2J_{\mu\nu} - K_{\mu\nu})$$

(数式17)

$$S_{IJ} = \int x_I(1) x_J(1) d\tau_1$$

(数式18)

$$P_{KL} = 2 \sum_{\nu} c_{K\nu} c_{L\nu}$$

(数式19)

$$G(I, J, K, L) = G(I, J, L, K) = G(J, I, K, L) = G(J, I, L, K) = \\ G(K, L, I, J) = G(L, K, I, J) = G(K, L, J, I) = G(L, K, J, I)$$

【図37】

(数式20)

$$X(r, n, R) = (r_x - R_x)^{n_x} (r_y - R_y)^{n_y} (r_z - R_z)^{n_z} \\ \times \sum_m d_m \exp[-\zeta_m (r - R)^2]$$

(数式21)

$$\phi(r, n, R) \\ = (r_x - R_x)^{n_x} (r_y - R_y)^{n_y} (r_z - R_z)^{n_z} \exp[-\zeta (r - R)^2]$$

(数式22)

$$\sum_m d_m \exp[-\zeta_m (r - R)^2]$$

(数式23)

$$G(I, J, K, L) = \sum_{m1} \sum_{m2} \sum_{m3} \sum_{m4} d_{m1} d_{m2} d_{m3} d_{m4} g(i, j, k, l)$$

(数式24)

$$g(i, j, k, l) = \int \int \phi_i(1) \phi_j(1) \frac{1}{r_{12}} \phi_k(2) \phi_l(2) d\tau_1 d\tau_2$$

(数式25)

$$g(i, j, k, l) = g(i, j, l, k) = g(j, i, k, l) = g(j, i, l, k) = \\ g(k, l, i, j) = g(l, k, i, j) = g(k, l, j, i) = g(l, k, j, i)$$

【図38】

(数式26)

$$\{0_a 0_b, 0_c 0_d\}^{(m)} \\ = \frac{1}{\sqrt{\zeta + \eta}} \cdot K(\zeta_a, \zeta_b, A, B) \cdot K(\zeta_c, \zeta_d, C, D) \cdot F_m(T)$$

(数式27)

$$F_m(T) = \int t^{2m} \exp(-T t^2) dt$$

(数式28)

$$T = \rho (P - Q)^2$$

(数式29)

$$K(\zeta, \zeta', R, R') \\ = \sqrt{2} \frac{\pi^{5/4}}{\zeta + \zeta'} \exp\left[-\frac{\zeta \zeta' (R - R')^2}{\zeta + \zeta'}\right]$$

(数式30)

$$\zeta = \zeta_a + \zeta_b$$

(数式31)

$$\eta = \zeta_c + \zeta_d$$

(数式32)

$$\rho = \frac{\zeta \eta}{\zeta + \eta}$$

(数式33)

$$P = \frac{\zeta_a A + \zeta_b B}{\zeta_a + \zeta_b}$$

(数式34)

$$Q = \frac{\zeta_c C + \zeta_d D}{\zeta_c + \zeta_d}$$



【図39】

(数式35)

$$\begin{aligned}
& [(a + 1_i) b, c d]^{(m)} \\
& = (P_i - A_i) [a b, c d]^{(m)} + (W_i - P_i) [a b, c d]^{(m+1)} \\
& \quad + \frac{N_i(a)}{2\zeta} \left\{ [(a - 1_i) b, c d]^{(m)} - \frac{\rho}{\zeta} [(a - 1_i) b, c d]^{(m+1)} \right\} \\
& \quad + \frac{N_i(b)}{2\zeta} \left\{ [a (b - 1_i), c d]^{(m)} - \frac{\rho}{\zeta} [a (b - 1_i), c d]^{(m+1)} \right\} \\
& \quad + \frac{N_i(c)}{(\zeta + \eta)} [a b, (c - 1_i) d]^{(m+1)} \\
& \quad + \frac{N_i(d)}{(\zeta + \eta)} [a b, c (d - 1_i)]^{(m+1)}
\end{aligned}$$

(数式36)

$$W = \frac{\zeta P + \eta Q}{\zeta + \eta}$$

(数式37)

$$\exp = \left[ -\frac{\zeta \zeta' (R - R')^2}{\zeta + \zeta'} \right]$$

(数式38)

$$\frac{R(R+1)}{2} + T$$

フロントページの続き

(72)発明者 稲畑 深二郎  
 神奈川県足柄上郡中井町境430 グリーン  
 テクナikai 富士ゼロックス株式会社内

(72)発明者 宮川 宣明  
 神奈川県足柄上郡中井町境430 グリーン  
 テクナikai 富士ゼロックス株式会社内

(72)発明者 高島 一  
 東京都豊島区高田3-24-1 大正製薬株  
 式会社内

(72)発明者 北村 一泰  
 東京都豊島区高田3-24-1 大正製薬株  
 式会社内

(72)発明者 長嶋 雲兵  
 東京都東久留米市金山町2-10-4

Fターム(参考) 5B045 GG02 GG15  
 5B098 AA10 GA03 GC08 GC16 GD02  
 GD14